

Angerhausen · Englisch · Gerits

***Tippek és trükkök
a Commodore
64-eshez***

DATA BECKER – NOVOTRADE

Angerhausen · Englisch · Gerits

Tippek és trükkök a Commodore 64-eshez

Csizm2722

DATA BECKER – NOVOTRADE

A mű eredeti címe: 64 Tips & Tricks (1983)

Fordította: DOBOSNÉ HARTYÁNYI MÁRIA

Szaklektorok: DR. OBÁDOVICS J. GYULA és DR. LENGYEL JÓZSEF

A kiadásért felel: RÉNYI GÁBOR, a NOVOTRADE RT. igazgatója
Budapest, 1985

Kiadványmenedzser: BÉKÉS TAMÁS
Sorozatszerkesztő: ROCHLITZ ANDRÁS
Felelős szerkesztő: TARR KÁLMÁNNÉ
Műszaki szerkesztő: DÉVÉNYI ERIKA

Szedte: az Alföldi Nyomda, Debrecen
Készült a Nyírségi Nyomdában, Nyíregyházán

ISBN 963 02 3985 X
Hungarian translation © Dobosné Hartyányi Mária

Copyright © 1983 DATA BECKER GmbH Merowingerstr. 30. 4000 Düsseldorf
Minden jog fenntartva. A DATA BECKER cég írásbeli hozzájárulása nélkül tilos a jelen könyvet
vagy annak részeit bármilyen eljárással (nyomtatás, fotókópia vagy egyéb technika), elektro-
nikus rendszerek felhasználásával másolni, sokszorosítani, terjeszteni.

FONTOS TUDNIVALÓ

A könyvben ismertetett kapcsolások, eljárások és programok nem tekintendők szabadalmi oltalom alá eső ipari termékeknek. Ezek elsősorban amatőr és oktatási célokat szolgálnak. A szerzők rendkívül nagy gondot fordítottak a kapcsolások, műszaki adatok és programok helyességére, a részletek kidolgozása során többszöri ellenőrzést végeztek. Mindez azonban nem zárja ki az esetleges hibalehetőségeket.

Az előforduló hibákért és az ebből adódó következményekért a DATA BECKER cég sem szavatosságot, sem jogi felelősséget nem vállal. Az esetlegesen előforduló hibák közlését a szerzők hálásan fogadják.

TARTALOMJEGYZÉK

1. FEJEZET

| | |
|--------------|---|
| Előszó | 9 |
|--------------|---|

2. FEJEZET

Grafika haladóknak

| | |
|---|----|
| 2.1 Grafika Commodore 64-esen | 11 |
| 2.1.1 A karakteres grafika | 11 |
| 2.1.2 A sprite-ok kezelése | 13 |
| 2.2 Háromdimenziós grafika – a 3D GRAFIKA BASIC program | 14 |
| 2.3 Színes sávgrafikonok | 17 |
| 2.4 Saját karakterkészlet | 21 |
| 2.5 A karakterkészlet módosítása joystickkal | 23 |
| 2.6 A billentyűzet-kiosztás és -módosítás | 27 |

3. FEJEZET

Kényelmes adatbevitel

| | |
|---|----|
| 3.1 A kurzor pozicionálása és a kurzorpozíció lekérdezése | 31 |
| 3.2 A kurzor ki- és bekapcsolása | 33 |
| 3.3 Az összes billentyű automatikus ismétlése (repeat) | 34 |
| 3.4 A WAIT utasítás: Várakozás egy billentyű megnyomására | 35 |
| 3.5 Egy kényelmes INPUT rutin | 36 |
| 3.6 A CBM 64-es „egere” | 40 |

4. FEJEZET

BASIC haladóknak

| | |
|---|----|
| 4.1 BASIC sor előállítása BASIC-ből | 45 |
| 4.2 A BASIC interpreter átmásolása a RAM-ba | 49 |
| 4.3 A negatív számok kiküszöbölése az FRE függvény értékkészletéből | 50 |
| 4.4 Visszatérés a BASIC programba LIST parancs után | 51 |
| 4.5 Változó sorszámok a GOTO, a GOSUB és a RESTORE utasításokban | 53 |
| 4.6 A MID\$ utasítás | 55 |
| 4.7 INSTR és STRING\$ függvények | 59 |
| 4.8 A Commodore 64 magyarul beszél | 63 |
| 4.9 A nyugat-berlini Ku'damm óra a CBM 64-esen | 66 |

5. FEJEZET

A Commodore nem csak a BASIC-et ismeri

| | |
|---|----|
| 5.1 Programozás FORTH nyelven | 71 |
| 5.2 A FORTH és a BASIC összehasonlítása | 73 |
| 5.3 További programnyelvek: PASCAL, LOGO, TURTLE GRAPHICS | 78 |

6. FEJEZET

A CP/M operációs rendszer a Commodore 64-esen

| | |
|---|-----|
| 6.1 Bevezetés a CP/M-be | 81 |
| 6.2 A CP/M rendszer programjai | 87 |
| 6.3 Standard CP/M szoftverek a CBM 64-esen | 90 |
| 6.4 A Z80-as processzor tárkiosztása | 90 |
| 6.5 Lemezkezelés CP/M alatt | 92 |
| 6.6 Együttműködés a 6510-es és a Z80-as processzorok között | 93 |
| 6.7 A CP/M BIOS listájának dokumentációja | 94 |
| 6.7.1 A BIOS 65 betöltőprogramja | 94 |
| 6.7.2 A BIOS 65 | 95 |
| 6.7.3 A Z80 BOOT-rutin | 100 |
| 6.7.4 A BIOS a Commodore 64-esen CP/M 2.2-re | 102 |
| 6.8 Saját INPUT/OUTPUT rutin beépítése a BIOS-ba | 116 |
| 6.9 Adatforgalom a CP/M programok és a Commodore BASIC között | 117 |

7. FEJEZET

Illesztési és bővítési lehetőségek a Commodore 64-esen

| | |
|--|-----|
| 7.1 Centronics nyomtató illesztése a számítógéphez | 121 |
| 7.2 A számítógépek közötti adatátvitel | 127 |
| 7.3 A CP/M cartridge illesztése az expansion porthoz | 131 |
| 7.4 Szintetizátor sztereóban | 134 |

8. FEJEZET

Adatfeldolgozás

| | |
|---|-----|
| 8.1 Kazettás egység – lemezegység | 139 |
| 8.2 Az adatfeldolgozás alapelve: a soros file | 143 |
| 8.3 Így gyorsabban megy: a relatív file-szervezés | 151 |
| 8.4 Egy másik módszer: a közvetlen hozzáférés | 161 |
| 8.5 Egy lezáratlan file megmentése | 163 |

9. FEJEZET

A POKE és más hasznos rutinok

| | |
|---|-----|
| 9.1 A kazettapuffer mint a program tárolására alkalmas hely | 167 |
| 9.2 A füzérek rendezése | 169 |
| 9.3 Az indexes változó legkisebb és legnagyobb értéke | 172 |
| 9.4 A DUMP utasítás – a változók tartalmának kiírása | 176 |
| 9.5 A módosított PEEK függvény | 179 |
| 9.6 Multiprogramozás a Commodore 64-esen | 181 |
| 9.7 A POKE utasítások és a nulláslap | 186 |

1. FEJEZET

ELŐSZÓ

A Commodore 64 naponta több ezernyi barátot szerez magának az egész világon. Ez a kis számítógépes rendszer nemcsak korlátlan lehetőséget biztosít a hobby-programozók számára, hanem minden további nélkül alkalmazható kisebb kereskedelmi, adminisztrációs és tudományos műszaki feladatok megoldásában is. Éppen ebben mutatkozik meg a „Tippek és Trükkök” c. könyv hasznossága. A Klaus Gerits, Lothar Englisch és Michael Angerhausen szerzői együttesünk ismét mélyrehatolt a lehetséges trükkök tanulmányozásában, miközben számtalan tippet adnak a C 64-es mind jobb kihasználásához. Az említett szerzők célkitűzése az volt, hogy példák és mintaprogramok alapján ismertessék a Commodore 64 alkalmazási lehetőségeit. A könyv nagy segítséget nyújt abban, hogy a Commodore 64 adottságait messzemenően kihasználhassuk.

Mindhárom szerző úgy ismeri a Commodore 64-est, mint a tenyerét. Klaus Gerits nevéhez fűződik a 64 IEC-BUS és a MAXI 64 kifejlesztése, Lothar Englisch pedig a PROFI-MON 64 és a PROFI-ASS 64 közismert programok alkotója. Mindketten behatóan ismerik a C 64-es műszaki és rendszertulajdonságait. Eredményeik közé tartozik a USER-port tervezetének kidolgozása és a CP/M-modul beható leírása, a minden részletre kiterjedően dokumentált BIOS-listával együtt. Michael Angerhausen – az adatbank-szakértőnk – a file-kezelés kapcsán foglalkozott behatóan a Commodore 64-essel. Többek között azt is bemutatja, hogy milyen eljárással lehet a BASIC 2.0. változat alatt a relatív file-okat felépíteni.

Néhány hónap alatt elfogyott a „Tippek és Trükkök” c. könyv első kiadása. Az új kiadást megelőzően a szerzők természetesen megragadták az alkalmat a közkedvelté és keresetté vált könyv átdolgozására és egyes részeinek kibővítésére. Olvasóink ezt a második, átdolgozott kiadást tarthatják a kezükben.

A „Tippek és Trükkök”-ben foglalt összes gépi programhoz és rutinhoz egy-egy BASIC betöltő programot (BASIC-Loadert) is mellékelünk. Ezeket az egyszerű BASIC programokat azok is könnyen betölthetik, akik gépi kódban nem tudnak programozni és így mindenki gond nélkül használhatja a gépi nyelven írt programokat.

A mindenkor BASIC betöltő ellenőrzi a gépi kódú programot, és kiszűri az adatbevitelből eredő hibákat.

Jó szórakozást kívánunk a „Tippek és Trükkök” c. könyvben foglalt feladatokhoz!

Düsseldorf, 1983. augusztus

A Szerzők

2. FEJEZET

GRAFIKA HALADÓKNAK

2.1 Grafika a Commodore 64-esen

A Commodore 64-es személyi számítógép tulajdonosai előbb-utóbb élni szeretnének a gépbe épített grafikus ábrázolási lehetőségekkel. A gép kézikönyve igen röviden foglalkozik a grafika-programozással.

A szerzők éppen ezért fontosnak tartották a grafikus ábrázolással kapcsolatos lehetőségek és sajátságok részletesebb ismertetését.

Mindenekelőtt különbséget kell tennünk a Commodore billentyűzetén keresztül elérhető karakteres grafika, a nagyfelbontású ábrázolás, ill. a sprite-grafika között. A karakteres-, ill. a nagyfelbontású színes grafika kezelésére több személyi számítógép képes, a sprite-ok ábrázolási lehetősége azonban kifejezetten a Commodore 64-es által kínált újdonság. A sprite-okat, melyek fantasztikus játékokra, kalandos történetek megjelenítésére, „Úr”-játékokra vagy támadó jellegű harci cselekmények szimulálására alkalmasak, eddig csak a játékautomaták kapcsán ismertük. A Commodore 64-es a sprite-grafika terén mindent tud, amit a játékautomaták valaha is tudtak!

Az Olvasó a könyv következő oldalain a három különböző grafikus ábrázolási lehetőséggel ismerkedhet meg. Természetesen az elméletet mindig szemléltető példákkal támasztjuk alá.

2.1.1 A karakteres grafika

A Commodore 64-esen ez a legegyszerűbb és legkényelmesebb grafikus ábrázolási mód, mert a grafikus karakterek megjelenítéséhez semmiféle tárcím kiszámítása nem szükséges, és nem kell a tár különböző rekeszeire figyelni. A megjelenítéshez mindig két billentyűt kell használnunk. A klaviatúrát alaposabban megvizsgálva látjuk, hogy minden egyes billentyűn két grafikus jel szerepel. Ha lenyomva tartjuk a C= (Commodore) billentyűt, és ezzel egyidejűleg leütünk egy tetszőleges billentyűt, a képernyőn megjelenik a billentyű bal oldalán látható grafikus karakter.

Ha programot írunk, a grafikus karaktereket a PRINT vagy az INPUT utasításokban helyezhetjük el. Ha például beírjuk a

```
100 PRINT"
```

utasítást, majd lenyomjuk egyidejűleg a C= és az A billentyűt, az idézőjel után egy olyan grafikus karakter jelenik meg, amelyet felhasználhatunk pl. egy keret bal felső sarkaként. Folytassuk a 100-as sort a következőképpen: lenyomjuk 38-szor a SHIFT és ezzel egyidejűleg a * billentyűt. (A billentyű jobb oldalán látható grafikus karaktert a SHIFT és az adott billentyű egyidejű leütésével jeleníthetjük meg a képernyőn.) Ha most ismét leütjük a C= ill. az S billentyűt, megkapjuk a keret jobb felső sarkát. Az utasítást egy idézőjellel, egy pontosvesszővel, végül a RETURN leütésével fejezzük be. A teljes keret előállításához még két sort kell begépelni.

110 PRINT"

Most üssük le 38-szor a SPACE billentyűt, majd ismét egy idézőjellel, pontosvesszővel, végül a RETURN leütésével fejezzük be a 110-es sort.
Az utolsó sor

120 PRINT"

majd ezután egyszer nyomjuk le a C= és a Z billentyűt egyidejűleg, majd 38-szor a SHIFT-et és a *-ot, s végül egyszer a C= és az X-et, s természetesen a RETURN leütésével fejezzük be a sort. Ezzel készen van a teljes keretet kirajzoló, 3 sorból álló program. Egészítsük ki az elkészült programot az alábbiakkal:

```
0 REM **** P1. ****
1
2
98 PRINT CHR$(147); REM A KÉPÉNYŐ TORLÉSE
100 PRINT "
110 PRINT "
120 PRINT "
130 A$=""
135 REM A$=98 ÜRES KARAKTER
140 B$="" **** EZ A SOR KITÖLTİ A KERETET ****
150 PRINT CHR$(19)
160 PRINT CHR$(17);CHR$(29);A$
170 PRINT CHR$(19)
180 PRINT CHR$(17);CHR$(29);B$
190 FOR I=1 TO 1000: NEXT
200 GOTO 150
```

Az ilyen keretek közkedveltek a kereskedelmi programokban, ugyanis áttekinthetővé teszik a képernyőt.

A grafikus karakterek megjelenítésére van egy másik lehetőség is. A programot a CHR\$ függvény felhasználásával is megírhatjuk, ha ismerjük a karakterek képernyőkódját:

```
0 REM **** P2. ****
1
2
100 A1$=CHR$(176);A2$=CHR$(174)
101 REM A BAL ES A JOBB FELSO SAROK
102 A3$=CHR$(173);A4$=CHR$(187)
103 REM A BAL ES A JOBB ALSO SAROK
104 H1$=CHR$(96)
105 REM A VÍZSZINTES VONAL
106 H2$=CHR$(125)
107 REM A FÜGGŐLEGES VONAL
108 H3$=CHR$(32)
109 REM AZ ÜRES KARAKTER
110 Z1$=A1$
111 FOR I=1 TO 38
112 Z1$=Z1$+H1$
113 NEXT I
114 Z1$=Z1$+A2$
115 REM A KERET 1. SORA
116 Z2$=H2$
117 FOR I=1 TO 38
118 Z2$=Z2$+H3$
119 NEXT I
120 Z2$=Z2$+H2$
```

```

121 REM A KERET 2. SORA
122 Z3$=A3$
123 FOR I=1 TO 38
124 Z3$=Z3$+H1$
125 NEXT I
126 Z3$=Z3$+A4$
127 REM A KERET 3. SORA
128 PRINT Z1$;
129 PRINT Z2$;
130 PRINT Z3$;

```

Ha lefuttatjuk ez utóbbi programot, azt tapasztaljuk, hogy a futás eredménye azonos az előzővel, előnye a könnyebb áttekinthetőség, és az egyszerű változtathatóság.

2.1.2 A sprite-ok kezelése

A Commodore 64-es jóval többre képes, mint a fentiekben ismertetett karakteres rajzolás. Ahogy azt már a korábbiakban említettük, grafikus lehetőségeit tekintve a C 64-es egyenértékű a nagy játékautomatákkal. Ebben a részben a teljesség igénye nélkül bemutatjuk a sprite-ok kezelésének programozási alapelveit.

A Commodore 64-es egyszerre nyolc különböző kis grafikus ábra – sprite – önálló kezelésére képes. A sprite-ok mindegyikét egyszerű POKE utasításokkal egymástól függetlenül mozgathatjuk, törölhetjük, egymással ütköztethetjük. Ez a felsorolás még korántsem meríti ki az összes lehetőséget. A programozónak természetesen ismernie kell a figurák mozgatását vezérlő tárcímeket, és azok jelentését. „A C 64 belső felépítése” című, ugyancsak a DATA BECKER cég által kiadott könyv a Commodore 64-es összes fontos tárcímét ismerteti. Itt most csak azokra térünk ki, amelyek a sprite-ok kezelésében szerepet játszanak.

A sprite-okra vonatkozó rekeszek az 53248-as címtől az 54271-es tárcímig terjedő tárterületen helyezkednek el. Ez a tárterület a gépbe beépített VIC 6569-es videovezérlő (video-controller) kommunikációs területe. Ha a képernyőn egy sprite-ot szeretnénk megjeleníteni, a kívánt képernyőpozíciót előzetesen közölnünk kell a VIC-kel. A kommunikációs terület egymást követő rekeszei egy-egy sprite helyzetének vízszintes, ill. függőleges koordinátáját határozzák meg.

Például a

```
POKE 53248,160:POKE 53249,120
```

utasításokkal egy sprite-ot a képernyő (160,120) koordinátájú pontjába helyezhetünk. A nyolc lehetséges sprite vízszintes, ill. függőleges koordinátáit az 53248-as tárcímtől kezdve rendre két-két cím tartalmazza. Ha a VIC regisztereinek számozását nullától kezdjük, az egyes sprite-hoz a 0 és 1, a kettes sprite-hoz a 2 és 3 számú rekeszek tartoznak stb.

* Eredeti címe: „64 Intern” (Data Becker, 1983).

Megjelenik magyarul a NOVOTRADE RT. kiadásában, 1985-ben. (A szerk.)

2.2 Háromdimenziós grafika – a 3D GRAFIKA BASIC program

Mindjárt a könyv elején ismerkedjünk meg egy olyan BASIC programmal, amely nagyfelbontású, színes grafikával háromdimenziós ábrákat jelenít meg a képernyőn. A program a Supergraphik 64 utasításait használja, de a szükséges változtatásokat közöljük arra az esetre is, ha az Olvasó nem rendelkezik a Supergraphik 64-gyel.

A P.5-ös program a 100-as programsorban megadott függvényt ábrázolja. Az ábrázolás háromféleképpen történhet:

- derékszögű (Descartes-féle) koordináta-rendszerben, mintha milliméterpapírra rajzolnánk;
- polár koordináta-rendszerben (a sugár és a szög megadásával);
- három dimenzióban.

Az utolsó – egyben a legérdekesebb – ábrázolásmóddhoz a függvényt el kell forgatni a függőleges Y-tengely körül. A kiszámítandó pontok nagy száma miatt ez az ábrázolási mód időigényes. A függvény képének paramétereit mindhárom esetben tetszőlegesen adhatjuk meg.

Érdeemes alaposan tanulmányozni a program működési elvét!

A program indításakor választhatunk a háromféle ábrázolási mód közül. Ha a derékszögű vagy a polár koordináta-rendszert választottuk, a program bekéri a lépésközt (260. sor). Ez az az érték, amellyel minden kiszámított pont után a vízszintes tengely paramétere növekszik. A 270. és a 280. sorban beolvassa az X- és az Y-tengely irányú nyújtási tényezőket. Ezzel meghatározhatjuk a tengelyek léptékviszonyát. Az első futtatásnál célszerű mindkét irányú nyújtásra egyet beírni. Hasonlóan, a vízszintes és függőleges irányú eltolás értékeiként (370–410. sor) először célszerű nullát beírni. A 430. sorban a program átkapcsol a nagyfelbontású grafikus üzemmódra. A 450–560. sorok rajzolják ki a koordináta-rendszert és a skálabeosztást, a koordináta-rendszer kiválasztásától függően a 680–790. sorok pedig kirajzolják a polár koordináta-rendszert, ill. a 820–970. sorok a derékszögű koordináta-rendszert. Az 1010. sortól kezdődik a háromdimenziós ábrázolás. Itt is megadhatjuk az ábrázolás léptékét és helyét: először célszerű 20 vagy 90 értéket beírni. Háromdimenziós ábrázoláshoz a 100. sorban a program több tízezer függvényérték-számítást végez, s ehhez a függvény típusától függően általában igen hosszú (fél órától több óráig terjedő) időre van szükség. Futtassuk le különböző függvényekkel a programot; érdemes például kipróbálni az alábbiakat:

```
0 REM **** P3. ****
1 :
2 :
100 DEF FN R(Q)=COS(2*Q)+COS((Q+BB)/16)
100 DEF FN R(Q)=SQR(ABS(.5*(16-Q*Q))+1/(Q+4)
100 DEF FN R(Q)=COS(4*Q)+20/(Q*Q+3)
```

Ha az Olvasó nem rendelkezik a „Supergraphik 64” segédprogrammal, az alábbi változtatásokat és kiegészítéseket kell megtenni:

```
0 REM **** P4. ****
1 :
2 :
5 POKE 56,32:CLR
430 FOR A1=8192 TO 16191:POKE A1,0:NEXT:GOSUB 2000
470 FOR A1=0 TO 199:AX=F:AY=A1:GOSUB 3000:NEXT
```

```

480 FOR A1=0 TO 319:AX=A1:AY=E:GOSUB 3000:NEXT
500 FOR A1=E-1 TO E+1:AX=XR:AY=A1:GOSUB 3000:NEXT
520 FOR A1=E-1 TO E+1:AX=XL:AY=A1:GOSUB 3000:NEXT
540 FOR A1=F-1 TO F+1:AX=A1:AY=YD:GOSUB 3000:NEXT
560 FOR A1=F-1 TO F+1:AX=A1:AY=YU:GOSUB 3000:NEXT
770 AX=XX:AY=YY:GOSUB 3000
900 AX=B:AY=YY:GOSUB 3000
1600 AX=X1:AY=Y1:GOSUB 3000
1620 GOSUB 4000:RETURN
2000 FOR A1=8192 TO 16191:POKE A1,0:NEXT
2010 FOR A1=1024 TO 2023:POKE A1,16:NEXT
2020 POKE 53248+17,27+32:POKE 53248+24,16+8
2030 RETURN
3000 OY=320*INT(AY/8)+(AYAND7)
3010 OX=8*INT(AX/8)
3020 MA=2^((7-AX)AND7)
3030 AV=8192+OY+OX
3040 POKE AV,PEEK(AV) OR MA:RETURN
4000 FOR A1=Y1+1 TO 199:AY=A1:GOSUB 5000:RETURN
5000 OY=320*INT(AY/8)+(AY AND 7)
5010 OX=8*INT(AX/8)
5020 MA=2^((7-AX) AND 7)
5030 AV=8192+OY+OX
5040 POKE AV,PEEK(AV) AND (255-MA):RETURN

```

A grafikus függvények BASIC nyelvű programozásával a program lényegesen lassúbb lesz, mint a „Supergraphik 64” segédprogram alkalmazásával.

```

0 REM **** P5. ****
1 :
2 :
10 PRINT "A SZAZAS SORBAN MEGHATAROZOTT"
20 PRINT " FUGGVENY GRAFIKUS ABRAZOLASA"
40 PRINT "1 - DEREKSZOGU KOORDINATARENDSZERBEN"
50 PRINT "2 - POLAR-KOORDINATARENDSZERBEN"
60 PRINT "3 - HAROM DIMENZIOBAN"
70 INPUT "VALASSZ: ";PL
100 DEF FNR(Q)=COS(Q)+COS(2*Q)+COS(3*Q)
210 IF PL=3 THEN 1010
250 PRINT:PRINT
260 INPUT "LEPESKOZ =";IK
270 INPUT "X-TENGELY NYUJTASI TENYEZOJE =";S1
280 INPUT "Y-TENGELY NYUJTASI TENYEZOJE =";S2
370 PRINT "ELTOLAS AZ X-TENGELY MENTEN"
380 INPUT "-130 ES 130 KOZE ESO SZAMERTEK ";C
400 PRINT "ELTOLAS AZ Y-TENGELY MENTEN"
410 INPUT "-90 ES 90 KOZE ESO SZAMERTEK ";D
430 IGCLEAR:IGRAPHICS 1:IBCOL 15:ILCOL 6
450 E=100:D:F=160+C
470 ILINE F,0 TO F,199
480 ILINE 0,E TO 310,E
490 FOR XR=F TO 319 STEP 19*S1
500 ILINE XR,E-1 TO XR,E+1 :NEXT
510 FOR XL=F TO 0 STEP-19*S1
520 ILINE XL,E-1 TO XL,E+1 :NEXT
530 FOR YD=E TO 199 STEP 15*S2
540 ILINE F-1,YD TO F+1,YD :NEXT
550 FOR YU=E TO 0 STEP -15*S2
560 ILINE F-1,YU TO F+1,YU :NEXT
580 IF PL=1 THEN 820
610 REM POLAR-KORD. PONT
690 RD=4/180:FOR G=0 TO 360 STEP IK :T=0*RD
710 X=FNR(T)*COS(T):Y=FNR(T)*SIN(T)
730 XX=X*(19*S1)+F:YY=Y*(15*S2)+E

```

```

740 IF XX<0 OR XX>319 THEN 780
750 IF YY<0 OR YY>199 THEN 780
770 !DOT XX,YY
780 NEXT
790 END
820 REM DEREKSZOGU KOORD. PONT
830 FOR G=0 TO 319 STEP 1K
840 X=(G-F)/(19*S1):Y=FNR(X)
850 YY=E-(Y*15*S2)
860 IF YY<0 OR YY>199 THEN 960
900 !DOT G,YY
960 NEXT
970 END
1010 REM HAROM DIMENZIOS PONT
1020 PRINT "FUGGOLEGES IRANYU LEPTEK"
1030 INPUT "40 ES 40 KOZE ES0 SZAM, JAVASOLT 20 "N1
1040 PRINT "FUGGOLEGES IRANYU FEKVES"
1050 INPUT "50 ES 150 KOZE ES0 SZAM, JAVASOLT 20 "N2
1260 REM KONSTANSOK A,B,C,D,E,F,G
1280 A=144:B=2.25:C=N1:D=.0327:E=160:F=N2:G=199
1400 !GRAPHICS! !GCLEAR!BCOL 15:LCOL 8
1410 FOR H=-A TO A STEP B
1420 AA=INT(.5+SQR(A*A-H*H))
1430 FOR BB=-AA TO AA !CC=SQR(BB*BB+H*H)*D
1440 D1=FNR(CC):DD=D1*C:GOSUB 1520:NEXT:NEXT:END
1450 GOTO 1450
1520 X=BB+H/B+E:Y=DD-H/B+F
1530 X1=INT(.85*X):Y1=INT(.9*(G-Y)):IF Y<0 OR Y>199 THEN RETURN
1600 !DOT X1,Y1
1620 !CLINE X1,Y1+1 TO X1,199:RETURN

```

2.3 Színes sávgrafikonok

A következő gépi kódú program vízszintes vagy függőleges színes sávgrafikonok rajzolására készült. Hasznos lehet, ha görbéket (függvényeket) vagy számtáblázatokat szeretnénk szemléletessé tenni. A grafikonokat képernyő-karakterekkel ábrázoljuk, így az ábrák mellett tetszőleges feliratokat is elhelyezhetünk. (Nagyfelbontású grafikus képernyőre szöveget nem lehet írni.)

A képernyőn vízszintesen 320, függőlegesen 200 pont áll rendelkezésre. A sávok elemi egységei egy karakter szélességűek, azaz nyolc vízszintes irányú képpontból állnak. A program a grafikon kirajzolását mindig az aktuális kurzorpozíción kezdi és minden függőleges, ill. vízszintes sávhoz megadható a magasság, ill. a hosszúság. A rutin felhasználásának megkönnyítésére egy-egy függőleges sáv kirajzolása után a kurzort automatikusan egy pozícióval jobbra mozgatjuk a képernyőn.

A rutin hívása az alábbi, kibővített SYS utasítással történhet:

SYS H, L, C vagy

SYS V, L, C

ahol a H és V a vízszintes, ill. függőleges grafikonok rutinjainak a kezdőcíme; L a grafikon hosszúsága (maximálisan 320), ill. magassága (maximum 200), C pedig a színkód (0-tól 15-ig).

A gépi kódú rutin listája:

```
0 REM **** P6. ****
1 :
2 :
100 : C000          .OPT P1
110 : C000          ; SZINES SAVGRAFIKON
120 : C000          ; HPLLOT ES VPLOT
130 : C000          GETCOR = $B7EB
140 : C000          SCROUT = $E716
150 : C000          LBYT = $14
160 : C000          HBYT = LBYT + 1
170 : C000          CURCOL = $D3
180 : C000          SETCOL = $EA24
190 : C000          SETCHAR = $EA1E
200 : C000          ILLQUAN = $B248
205 : C000          CHKCOM = $AEFD
210 : C000          CODE = $22
220 : C000          TMP = CODE + 1
230 : C000          XREG = TMP + 1
235 : C000          TMP1 = XREG + 1
240 : C000          FARB = $F3 ; MUTATO A SZIN-RAM-RA
250 : C000          CURRIGHT = $AB3B
260 : C000          ADR = $FD
270 : C000          LINELEN = $D5
280 : C000          CHARADR = $D1
290 : C000          *= $C000
300 : C000 20 FD AE HPLLOT JSR CHKCOM ; VESSZO
310 : C003 20 EB B7 JSR GETCOR
315 : C006 B6 24 STX XREG
320 : C008 A5 15 LDA HBYT
330 : C00A C9 02 CMP #2
340 : C00C B0 42 BCS ILL
350 : C00E 0A ASL
360 : C00F 0A ASL
370 : C010 0A ASL
```

```

380 : C011 0A      ASL
390 : C012 0A      ASL
400 : C013 85 23   STA TMP
410 : C015 A5 14   LDA LBYT
420 : C017 4B      PHA
430 : C018 4A      LSR
440 : C019 4A      LSR
450 : C01A 4A      LSR
460 : C01B 1B      CLC
470 : C01C 65 23   ADC TMP
480 : C01E 65 D3   ADC CURCOL
490 : C020 4B      PHA
500 : C021 A8      TAY
510 : C022 C5 D3   CMP CURCOL
520 : C024 F0 13   BEQ T1
530 : C026 C9 27   CMP #39
540 : C028 90 02   BCC T2
550 : C02A A0 27   LDY #39
560 : C02C 20 24 EA T2 JSR SETCOL
570 : C02F A9 A0   LDA # " + $80
580 : C031 20 1E EA JSR SETCHAR
590 : C034 8B      DEY
600 : C035 C4 D3   CPY CURCOL
610 : C037 10 F3   BPL T2
620 : C039 6B      PLA
630 : C03A A8      TAY
640 : C03B 6B      PLA
650 : C03C C0 2B   CDY #48
660 : C03E B0 0B   BCS FERTIG
670 : C040 29 07   AND #7
680 : C042 AA      TAX
690 : C043 B0 53 C0 LDA TABLE,X
700 : C046 A6 24   LDX XREG
710 : C048 20 1E EA JSR SETCHAR
720 : C04B A9 11   LDA #17
730 : C04D 4C 16 E7 JMP SCROUT
740 : C050 4C 4B B2 IL JMP ILLQUAN
750 : C053 20 65 74 TABLE .BYT $20,$65,$74,$75,$61,$F6,$EA,$E7
1000 : C05B 20 FD AE VPL0T JSR CHKCDM
1010 : C05E 20 EB B7 JSR GETCOR
1020 : C061 A5 15   LDA HBYT
1030 : C063 D0 EB   BNE ILL
1040 : C065 86 24   STX XREG
1050 : C067 A5 14   LDA LBYT
1060 : C069 4A      LSR
1070 : C06A 4A      LSR
1080 : C06B 4A      LSR
1090 : C06C 85 23   STA TMP
1100 : C06E A5 14   LDA LBYT
1110 : C070 29 07   AND #7
1120 : C072 85 25   STA TMP1
1130 : C074 A5 D1   LDA CHARADR
1140 : C076 1B      CLC
1150 : C077 65 D3   ADC CURCOL
1160 : C079 85 FD   STA ADR
1170 : C07B A5 D2   LDA CHARADR + 1
1180 : C07D 69 00   ADC #0
1190 : C07F 85 FE   STA ADR+1
1200 : C081 A0 00   LDY #0
1210 : C083 A6 23   LDX TMP
1220 : C085 F0 20   BEQ T3
1230 : C087 20 C7 C0 T4 JSR COLOR
1240 : C08A A9 A0   LDA # " + $80
1250 : C08C 91 FD   STA (ADR),Y
1260 : C08E A5 24   LDA XREG
1270 : C090 91 F3   STA (FARB),Y
1280 : C092 A5 FD   LDA ADR

```


| | | |
|---------------------------|--|------------------------------------|
| 1290 : C094 38 | SEC | |
| 1300 : C095 E9 28 | SBC #40 | ; A KOVETKEZO SOR |
| 1310 : C097 85 FD | STA ADR | |
| 1320 : C099 80 08 | BCS T5 | |
| 1330 : C09B C6 FE | DEC ADR+1 | |
| 1340 : C09D A5 FE | LDA ADR+1 | |
| 1350 : C09F C9 04 | CMP #4 | ; A LEGFELSO SOR |
| 1360 : C0A1 90 12 | BCC T6 | |
| 1370 : C0A3 C6 23 T5 | DEC TMP | |
| 1380 : C0A5 D0 E0 | BNE T4 | |
| 1390 : C0A7 20 C7 C0 T3 | JSR COLOR | |
| 1400 : C0AA A6 25 | LDX TMP1 | |
| 1410 : C0AC BD BF C0 | LDA TAB2,X | |
| 1420 : C0AF 91 FD | STA (ADR),Y | |
| 1430 : C0B1 A5 24 | LDA XREG | ; SZIN |
| 1440 : C0B3 91 F3 | STA (FARB),Y | |
| 1450 : C0B5 A5 D3 T6 | LDA CURCOL | |
| 1460 : C0B7 C5 D5 | CMP LINELN | ; A KURZOR AZ UTULSO OSZLOPBAN VAN |
| 1470 : C0B9 F0 03 | BEQ * + 5 | |
| 1480 : C0BB 4C 3B AB | JMP CURRIGHT | ; KURZOR JOBBRA |
| 1490 : C0BE 60 | RTS | |
| 1500 : C0BF 20 64 6F TAB2 | .BYT \$20,\$64,\$6F,\$79,\$62,\$F8,\$F7,\$E3 | |
| 1510 : C0C7 A5 FD COLOR | LDA ADR | |
| 1520 : C0C9 85 F3 | STA FARB | |
| 1530 : C0CB A5 FE | LDA ADR+1 | |
| 1540 : C0CD 4C 2A EA | JMP SETCOL + 6 | |

Azoknak, akik nem rendelkeznek assembler vagy monitor segédprogramokkal, közöljük a BASIC betöltőprogramot:

```

0 REM **** P7. ****
1 :
2 :
100 FOR I=49152 TO 49359
110 READ X:POKE I, X:S=S+X:NEXT
120 DATA 32,253,174, 32,235,183,134, 36,165, 21,201, 2
130 DATA 176, 66, 10, 10, 10, 10,133, 35,165, 20, 72
140 DATA 74, 74, 74, 24,101, 35,101,211, 72,168,197,211
150 DATA 240, 19,201, 39,144, 2,160, 39, 32, 36,234,169
160 DATA 160, 32, 30,234,136,196,211, 16,243,104,168,104
170 DATA 192, 40,176, 11, 41, 7,170,189, 83,192,166, 36
180 DATA 32, 30,234,169, 17, 76, 22,231, 76, 72,178, 32
190 DATA 101,116,117, 97,246,234,231, 32,253,174, 32,235
200 DATA 183,165, 21,208,235,134, 36,165, 20, 74, 74, 74
210 DATA 133, 35,165, 20, 41, 7,133, 37,165,209, 24,101
220 DATA 211,133,253,165,210,105, 0,133,254,160, 0,166
230 DATA 35,240, 32, 32,199,192,169,160,145,253,165, 36
240 DATA 145,243,165,253, 56,233, 40,133,253,176, 8,198
250 DATA 254,165,254,201, 4,144, 18,198, 35,208,224, 32
260 DATA 199,192,166, 37,189,191,192,145,253,165, 36,145
270 DATA 243,165,211,197,213,240, 3, 76, 59,171, 96, 32
280 DATA 100,111,121, 98,248,247,227,165,253,133,243,165
290 DATA 254, 76, 42,234
300 IF S<>26696 THEN PRINT"HIBA A DATA SORBAN..!":END
310 PRINT"OK.!"

```

Az alábbiakban egy példát mutatunk a sávgrafikonokat készítő gépi kódú rutin felhasználására. Első mintafeladatunkban egy forgalmi statisztikát ábrázolunk szemléletes oszlopdiagramokkal.

```

0 REM **** P8. ****
1 :
2 :
3 POKE 53280,0:POKE 53281,0
100 REM AZ EGY EVRE VONATKOZO HAVI OSSZEGEK
110 REM ADATSORBAN VANNAK ELHELVEZVE.
120 DIM U(12)
130 REM AZ ADATOK BEOLVASASA
140 FOR I=1 TO 12:READ U(I):NEXT I
150 REM A MAXIMALIS ERTEK MEGHATAROZASA
160 MAX=0
170 FOR I=1 TO 12
180 IF U(I)>MAX THEN MAX=U(I)
190 NEXT I
200 V=12*4096+5*16+11:REM A BEPIKODU SZUBROUTIN CIME
210 PRINT CHR$(147):REM KEPERNYO TORLES
220 FOR I=1 TO 21:PRINT CHR$(17);:NEXT I:REM CURSOR
230 REM A SAVGRAFIKON KIRAJZOLASA
240 FOR I=1 TO 12
250 PRINT SPC(2);:SYS V,U(I)/MAX*180,I
260 NEXT I
270 PRINT:PRINT
280 REM HONAPOK KIIRASA
290 FOR I=1 TO 12
300 PRINT RIGHT$(" "+STR$(I),3);
310 NEXT I
320 GET A$:IF A$="" THEN 320
330 REM HAVI OSSZEGEK
340 DATA 12000,13500,11000, 8000,14000, 9000
350 DATA 13800,14000,12750,14000,13800,17200

```

Most ábrázoljunk egy függvényt vízszintes sávgrafikonokkal.

A 100. sorban a program beolvassa a színkódot. Ezután töröljük a képernyőt, a háttér feketére vált és a kurzor átkerül a második oszlopba. A 120. és a 130. sorban kiszámítjuk a függvényértékeket a -2.2 -től 2.2 -ig terjedő intervallumban, a kapott értékeket az arányos ábrázolás érdekében beszorozzuk 300 -zal, és végül a `SYS` utasítással meghívjuk a rutint, amely a függvényt kirajzolja.

```

0 REM **** P9. ****
1 :
2 :
100 INPUT "SZIN";C:IF C<1 OR C>15 THEN 100
110 H=12*4096:PRINT CHR$(147) TAB(2);:POKE 53281,0
120 FOR I=-2.2 TO 2.2 STEP .2
130 SYS H,EXP(-I*I)*300,C:NEXT I

```

2.4 Saját karakterkészlet

A Commodore 64-es egyik hasznos sajátossága, hogy karaktergenerátora áthelyezhető a RAM-ba. Minthogy a RAM-ot a gép használója igénye szerint átalakíthatja – a karaktergenerátort áthelyezve –, ezen a tárterületen saját karakterkészletet határozhatunk meg. Az operációs rendszer a karaktereket egy-egy 8×8 pontból álló karaktermátrix segítségével ábrázolja. A tárban minden karaktermátrixhoz 64 bit, azaz 8 byte tartozik. Ha valamely bit magas (azaz értéke 1), a hozzárendelt mátrixpont látható a karakterben, egyébként nem.

Az alábbi kis program segítségével a gép által használt karakterkészlet egy elemét megjeleníthetjük a képernyőn. Mivel a program a 9.5 fejezetben ismertetett módosított PEEK függvény-nel dolgozik, használatát előtt be kell tölteni a 9.5 fejezetben közölt gépi kódú programot.

```
0 REM **** P10. ****
1 :
2 :
100 PRINT CHR$(147):PRINT:PRINT:PRINT
110 INPUT "KEREM A KARAKERT ";A$
120 PRINT CHR$(19)A$:C=PEEK(1024)
130 PRINT:PRINT:PRINT:PRINT
140 CG=13*4096:REM A KARAKTERGENERATOR KEZDO CIME
150 REM KIS/NAGYBETU UZEMMOD-GRAFIKUS UZEMMOD
160 B=(PEEK(53248+24)AND2)*1024
170 FOR I=0 TO 7
180 Z=USR(CG+B+8*C+I):REM A MATRIX BETOLTESE SORONKENT
190 FOR J=7 TO 0 STEP -1
200 A=Z AND 2^J
210 IF A THEN PRINT "*";:GOTO 230
220 PRINT " ";
230 NEXT
240 PRINT
250 NEXT
255 FOR I=1 TO 5000:NEXT
260 RUN
```

A program bekéri a karaktert, amelynek mátrixát a képernyőn szeretnénk megjeleníteni. A karakter képernyőkódját meghatározza a képernyőtárból és beviszi a B változóba (120. sor). A 140. sorban megvizsgálja, hogy melyik (nagybetűs/kisbetűs vagy nagybetűs/grafikus) üzemmódban használjuk a billentyűzetet. A 160. sorban meghatározza a karaktermátrix kezdőpozícióját a karaktergenerátorban.

A 180. sorban megvizsgálja, hogy egy mátrixpontnak megfelelő bit magas vagy alacsony. Ettől függően egy csillagot vagy egy pontot kirajzol a képernyőre. Ha futtatjuk a programot, és például a T betű kirajzolását kérjük, a képernyőn a következő ábrát kapjuk:

```
*****
- - - * * - - -
- - - * * - - -
- - - * * - - -
- - - * * - - -
- - - * * - - -
- - - * * - - -
- - - * * - - -
- - - - - =
```

Most nézzük meg, hogy hogyan lehet BASIC programmal saját karaktert előállítani. Először át kell helyoznunk a karaktergenerátort a ROM-ból a RAM-ba, majd közölnünk kell az operációs rendszerrel az áthelyezett karaktergenerátor tárbeli kezdőcímét. Ezzel egyidejűleg a képernyőtárat is át kell helyoznunk a \$C400 kezdőcímre (decimálisan: az 50176-tól az 51175-ig terjedő tárterületre). Az áthelyezést POKE ciklussal végezzük el, amelyhez ismét felhasználjuk a 9.5 fejezetbeli USR függvényt.

```
0 REM **** P11. ****
1 :
2 :
100 FOR I=13*4096 TO 14*4096-1
110 POKE I+4096,USR(I):NEXT I
120 POKE 53272,24:POKE 56576,148:POKE 648,196
```

A fenti kis program végrehajtása után következhet a saját karakter meghatározása a P12-es programmal. A program bekéri a módosítandó karaktert, majd megadhatjuk az új karakter mátrixát a billentyűzetről úgy, hogy minden leütött csillag látható (bekapcsolt) karakterpontot, minden leütött pont pedig nem látható (kikapcsolt) karakterpontot jelent. Tetszőleges számú saját karakter meghatározása után a VÉGE szó beütésével jelezhetjük, hogy a karakterkészlet módosítását befejeztük.

```
0 REM **** P12. ****
1 :
2 :
100 REM KARAKTEREK KESZITESE
110 CG=14*4096:REM KARAKTER GENERATOR KEZDOCIME
200 INPUT "MELYIK KARAKTERT KIVANJA ATIRNI "A$:IF A$="VEGE" THEN END
210 PRINT "":A$
220 C=PEEK(12*4096+1024)
230 PRINT "C=1234567"
300 FOR I=0 TO 7
310 PRINT I:INPUT A$(I):IF LEN(A$(I))<>8 THEN PRINT "XX":GOTO 310
320 NEXT
400 B=(PEEK(53248+24)AND2)*1024:REM NAGYBETU/GRAFIKA UZEMMOD
405 AD=CG+B+C*8
410 FOR I=0 TO 7:2=0
420 FOR J=0 TO 7
430 Z=Z-(MID$(A$(I),8-J,1)="*")*2+J
440 NEXT
450 POKE AD+I,Z:REM KARAKTER
460 POKE AD+1024+I,255-Z:REM RVB KARAKTER
470 NEXT
480 GOTO 200
```

2.5 A karakterkészlet módosítása joystickkal

Ha gyakran dolgozunk programjainkban különleges karakterekkel (például egy matematikai programban görög betűkkel), a kívánt karakterek létrehozásának előzőekben ismertetett módja meglehetősen lassú és kényelmetlen. Különösen akkor, ha nemcsak betűket, hanem kis grafikus figurákat is szeretnénk felhasználni. Legjobb megoldás, ha előzetesen raszterpáíron megtervezzük a figurát, kiszámítjuk a pontsorokhoz tartozó bitkombinációkat, majd a bináris szám decimális megfelelőjét POKE utasítással byte-onként elhelyezzük a RAM-beli karaktergenerátorban. Ez a munka meglehetősen lassú és fáradságos. A kényelmetlenségek elkerülésére bemutatunk egy olyan programot, amellyel egy karaktert felnagyítva ábrázolhatunk és a joystickkal (botkormánnyal) tetszőlegesen módosíthatjuk a képernyőn. A botkormányt a program az 1-es portról használja).

Az új karakterkészlet kialakításához két, a RAM-ban elhelyezett karaktergenerátorra van szükség, azaz az eredeti ROM-beli karaktergenerátorból két másolatot kell képezni. A program a módosítandó karaktert sprite-ként ábrázolva megjeleníti a képernyőn eredeti és kétszeres nagyságban, és belehelyez egy villogó mikrokurzort, amelyet a joystickkal tetszőleges irányba elmozdíthatunk. A joystick tűzgombjával vezérelhetjük, hogy milyen műveletet akarunk végezni (vonalat húzni, törölni, pozicionálni stb.).

Ha elégedettek vagyunk a megrajzolt karakterrel, az F1 gombot kell megnyomnunk, mire a program a karaktergenerátor második másolatában a módosító karakter helyére betölti az új karaktert. Azt, hogy egy karaktert változatlanul hagyunk, a G billentyű megnyomásával jelezhetjük.

A program az 512 karakterből álló teljes karakterkészlet feldolgozása után véget ér.

Az 512 karakter egy kicsit soknak tűnik, de ha figyelembe vesszük a billentyűzet mindkét üzemmódját (nagybetűs/grafikus, ill. nagybetűs/kisbetűs), üzemmódonként 128 lehetséges karakterrel, és ezt megduplázzuk – a fordított (reverse) ábrázolásmód miatt –, az eredmény tényleg pontosan 512.

Az alábbiakban részletesen ismertetjük a programban használt változókat és a program működését.

A változólista:

| | |
|----|---|
| C | A soron következő karakter báziscíme az első másolatban. |
| CD | A karakter báziscíme a második másolatban. |
| CP | 0–511 tartományú karakterpozíció számláló. |
| JB | A tűzgomb állapota. |
| JR | A joystick kapcsolóállása. |
| JS | Az 1-es port címe |
| MA | Üzemmód-kapcsoló |
| PO | A mikrokurzor pozíciója a megcímezett byte-on belül. |
| PP | A mikrokurzor pillanatnyi helyzetének megfelelő cím a sprite-on belül. |
| PV | A botkormány pillanatnyi állásának megfelelő byte értéke a sprite-on belül. |
| SB | A sprite adatok báziscíme. |
| V | A videovezérlő báziscíme. |
| X | A sprite x-pozíciója a képernyőn. |
| XJ | A botkormánnyal vezérelt mikrokurzor x-pozíciója. |
| Y | A sprite-ok y-pozíciója a képernyőn. |
| YJ | A botkormány által vezérelt mikrokurzor y-pozíciója. |

Érintett tárcímek:

| | |
|-------|---|
| 56 | A RAM-top mutató felső byte-ja |
| 648 | A videoram kezdete (mutató) |
| 832 | A kazettapuffer kezdőcíme. (Minthogy erre a területre nincs szükségünk, ide helyeztük a gépi kódú programot.) |
| 50169 | Az első sprite kezdete (mutató) |
| 50170 | A második sprite kezdete (mutató) |
| 53272 | A videoram és a karaktergenerátor kezdete a videovezérlőn belül. |
| 56576 | A képernyővel összefüggő 16 kbyte-os tárterületet kiválasztó biteket tartalmazó tárcím |

A program dokumentációja:

| | |
|-----------|--|
| 10 | A tár felső határát eltoljuk, ugyanis ide kell elhelyezni a karakterkészlet első másolatát. |
| 30233 | Bekapcsoljuk az első és a második sprite-ot és beállítjuk a színüket. Betöltjük a sprite-mutatókat, majd a második sprite-ot kétszeresére nagyítjuk. Töröljük a sprite-adatokat, majd a sprite-okat a képernyő közepére pozicionáljuk. |
| 1000–1010 | A gépi kódú rutin DATA sora |

A gépi kódú rutin assembler utasításai:

| | |
|------------|-----------------------------------|
| sei | A megszakítás letiltása |
| lda #\$33 | A karaktergenerátor megnyitása |
| sta 1 | |
| lda #0 | |
| sta \$5f | A régi blokk-kezdet (alsó byte) |
| sta \$5a | A régi blokkvég (alsó byte) |
| sta \$58 | Az új blokkvég (alsó byte) |
| lda \$d0 | |
| sta \$60 | A régi blokk-kezdet (felső byte). |
| lda #\$10 | |
| sta \$59 | Az új blokkvég (felső byte). |
| lda #\$e0 | |
| sta \$5b | A régi blokkvég (felső byte). |
| jsr \$a3bf | A blokkokat áthelyező rutin |
| lda #\$37 | |
| sta 1 | |
| cli | A megszakítás engedélyezése |
| ret | Ugrás vissza a BASIC-hez |

| | |
|-----------|---|
| 1020–1040 | A gépi program betöltése a kazettapufferbe és futtatása. Futás után a karaktergenerátorról elkészül a két másolat. |
| 1060 | A program közli az operációs rendszerrel a változásokat. A megváltozott karaktergenerátor és a videoram helyzete (az utóbbi hardver okok miatt!) |

- 2000-2360 A képernyőn most látható karakterek már a második másolatból származnak.
 2380 A karakterek egymás utáni áttöltése a sprite-adatokba és módosításuk.
 Az összes karakter kezelése után a RAM-top helyreállítása. (Az első másolatra már nincs szükség.)
 4000-4070 Lekérdezzük a botkormányt, majd a botkormány és a tűzgomb állásától függően elágaztatjuk a programot.
 5000-12040 A joystick állásától függően aktualizáljuk a kurzorpozíciót.
 13000-13200 A botkormány pozícióját átszámítjuk a mikrokurzor pozíciójára, majd ezt a pontot felváltva be- és kikapcsoljuk.
 20000-20080 A kész karaktereket áttöltjük a második másolatba.

A módosított karaktergenerátor nem foglal el BASIC tárterületet, a RAM-ban az operációs rendszer alatt helyezkedik el. (A Commodore 64-es sajátossága a RAM és ROM közötti átlapolási lehetőség.)

A videoram kezdőcíme a 49152, amit mindig szem előtt kell tartani, ha a tárcímek tartalmát POKE utasítással módosítjuk. A gép hardver-felépítéséből következik, hogy a karaktergenerátor áthelyezésekor a videoram is eltolódik. Ezzel a problémakörrel részletesen „A C 64 belső felépítése” c. könyvből ismerkedhetünk meg.

A program listája:

```
0 REM **** P13. ****
1 :
2 :
10 POKE 56,144:CLR
20 V=53248
30 POKE V+21,6:POKE V+40,1:POKE V+41,1
40 POKE 50169,16:POKE 50170,16
42 POKE V+23,4:POKE V+29,4
50 FOR I=0 TO 62:POKE 50176+I,0:NEXT
55 X=150:Y=100
223 POKE V+4,X
226 POKE V+2,X-40
233 POKE V+16,0
320 POKE V+5,Y:POKE V+3,Y+19
1000 DATA 120,169,51,133,1,169,0,133,95,133,90,133,88,169,208,133,96,169,240
1010 DATA 133,89,169,224,133,91,32,191,163,169,55,133,1,88,96
1020 FOR I= 832 TO 832+33
1030 READ A:POKE I,A:NEXT
1040 SYS 832:POKE 850,160:SYS 832
1060 POKE 53272,8:POKE 56576,PEEK(56576)AND252:POKE 648,192
1070 PRINT CHR$(179)
2000 C=9*4096
2020 FOR CP=0 TO 511:PRINT CHR$(19)CP:SB=50176
2040 FOR I=0 TO 7
2060 POKE SB+3*I,PEEK(C+I)
2080 NEXT I
2360 C=C+8:GOSUB 4000:NEXT CP
2380 POKE V+21,0:POKE 56,160:CLR:END
4000 XJ=0:YJ=0:JS=56321:SB=50176
4020 JR=(255-PEEK(JS)) AND 15:JB=(255-PEEK(JS)) AND 16
4030 IF JB THEN MA=MA+1:IF MA>2 THEN MA=0
4040 ON JR GOTO 5000,6000,4020,7000,8000,9000,4020,10000,11000,12000
4045 IF PEEK(203)<>4 THEN 4066
4050 PRINT CHR$(147)CHR$(51)CHR$(17)CHR$(17)CHR$(17)CHR$(17)CHR$(18) " MENTES " : B
OSUB 20000
4055 PRINT CHR$(147)CHR$(51)CHR$(17)CHR$(17)CHR$(17)CHR$(17):RETURN
4066 IF MA=1 THEN PRINT CHR$(147)CHR$(51)CHR$(17)CHR$(17)CHR$(18) " ALLITAS"
4067 IF MA=2 THEN PRINT CHR$(147)CHR$(51)CHR$(17)CHR$(17)CHR$(18) " TORLES "
4068 IF PEEK(203)=26 THEN RETURN
```

```

4069 IF MA=0 THEN PRINT CHR$(147)CHR$(51)CHR$(17)CHR$(17)"
4070 GOSUB 13000:GOTO 4020
5000 REM FOLSO
5020 YJ=YJ-1:IF YJ<0 THEN YJ=0
5040 GOSUB 13000:GOTO 4020
6000 REM ALSO
6020 YJ=YJ+1:IF YJ>7 THEN YJ=7
6040 GOSUB 13000:GOTO 4020
7000 REM BAL
7020 XJ=XJ-1:IF XJ<0 THEN XJ=0
7040 GOSUB 13000:GOTO 4020
8000 REM BAL FOLSO
8020 XJ=XJ-1:IF XJ<0 THEN XJ=0
8040 GOTO 5000
9000 REM BAL ALSO
9020 XJ=XJ-1:IF XJ<0 THEN XJ=0
9040 GOTO 6000
10000 REM JOBB
10020 XJ=XJ+1:IF XJ>7 THEN XJ=7
10040 GOSUB 13000:GOTO 4020
11000 REM JOBB FOLSO
11020 XJ=XJ+1:IF XJ>7 THEN XJ=7
11040 GOTO 5000
12000 REM JOBB ALSO
12020 XJ=XJ+1:IF XJ>7 THEN XJ=7
12040 GOTO 6000
13000 REM
13020 PP=SB+YJ*3+INT(XJ/8):PV=PEEK(PP)
13040 PO=XJ-INT(XJ/8)*8
13060 IF PV AND 2^(7-PO) THEN POKE PP,(PV AND (255-2^(7-PO))):GOTO 13100
13080 POKE PP,(PV OR (2^(7-PO)))
13100 IF MA=1 THEN PV=(PV OR (2^(7-PO)))
13120 IF MA=2 THEN PV=(PV AND (255-2^(7-PO)))
13200 FOR I=0 TO 50:NEXT:POKE PP,PV:RETURN
20000 REM AZ UJ KARAKTER ATVITELE
20010 CD=C+20472
20020 FOR I=0 TO 7
20040 POKE CD+I,PEEK(SB+3*I)
20060 NEXT I
20080 RETURN

```


2.6 A billentyűzet-kiosztás és -módosítás

A Commodore 64-es billentyűzete mátrixszerűen nyolc sorban és nyolc oszlopban rendezhető el. A nyolc sorhoz tartozó vezetékek a CIA 1 A portjához (\$DC00=56320 cím), a nyolc oszlophoz tartozók pedig a CIA 1 B portjához (\$DC01=56321 cím) csatlakoznak. A billentyűzet lekérdezései (\$FF9F=65485 cím) az A porton minden sorról egy-egy jel érkezik. Ha egy billentyű le van nyomva, a B porton keresztül lekérdezhetjük az oszlopot. A sor és az oszlop számából a rendszer kiszámítja egy 0 és 63 közé eső ún. billentyűszámot. Ha ennek értéke 64, akkor egyetlen billentyű sincs lenyomva (a hozzárendelést alább szemléltetjük). A billentyűszámot a rendszer minden lekérdezés után \$CB (203) címen tárolja; az utolsóként lenyomott billentyű száma pedig a \$C5 (197) címre kerül. A különleges billentyűk állapotát a \$028D (653) cím tartalma jelzi. A nullás bit a SHIFT, az egyes bit a Commodore, a kettes bit pedig a CTRL billentyűre vonatkozik. A karakterek és a billentyűk egymáshoz rendelését az ASCII kódokat tartalmazó tárbeli táblázatokkal oldották meg. Mivel a Commodore 64-es minden billentyűje több funkciót lát el, összesen 4 ilyen kódtáblázat van a tárban.

A billentyűzetet szemléltető ábra elemzésekor ne feledkezzünk meg arról, hogy a bal és jobb oldali SHIFT billentyűk nem egyenértékűek. A Shift Lock (zár) funkciót a bal oldali SHIFT billentyű látja el. (A legújabb típusú alapgépeken a SHIFT LOCK billentyű külön van.)

| Oszlop | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|--------|------|--------|----------|------|-----------------|----|----|----------------|
| Sor | | | | | | | | |
| 0 | DEL | RETURN | CURRIGHT | F7 | F1 | F3 | F5 | CURDOWN |
| 1 | 3 | W | A | 4 | Z | S | E | SHIFT (BAL) |
| 2 | 5 | R | D | 6 | C | F | T | X |
| 3 | 7 | Y | G | 8 | B | H | U | V |
| 4 | 9 | I | J | 0 | M | K | O | N |
| 5 | + | P | L | — | . | : | ⌂ |) |
| 6 | FONT | | ; | HOME | SHIFT (JOBB) | = | ↑ | / |
| 7 | 1 | PFL | CTRL | 2 | SPACE | C= | Q | STOP |

Az 1. tárbeli kódtáblázat az önállóan lenyomott billentyűk karaktereinek ASCII kódjait tartalmazza. A 2., 3. és 4. táblázat rendre az adott billentyű, ill. a SHIFT, a C= és a CTRL billentyűk együttes lenyomásával elérhető karakterek ASCII kódjaiból épül fel.

A táblázatokban előforduló \$FF (255) érték a nem létező kombinációkra utal. A SHIFT, a C= és a CTRL billentyűket külön kell kezelni, az ezeknek megfelelő érték az 1. táblázatban 1, 2 és 4. A különleges billentyűk állapotát a \$28D (653) tárcím 0., 1. és 2. bitje jelzi.

Ha valamelyik billentyűhöz egy másik kódot akarunk hozzárendelni, a megfelelő bejegyzést a kódtáblázatban módosítanunk kell. Mivel a táblázatok a ROM-ban vannak, a bejegyzéseket nem lehet egyszerűen felülírni. Hasonlóan járhatunk el, mint a saját karakterkészlet kialakításánál, az operációs rendszert átmásoljuk az „alatta lévő” RAM-ba, majd itt végrehajtjuk a kívánt változtatást. A tárkiosztás miatt eközben a BASIC ROM-ot is át kell helyeznünk a RAM területre. Az alábbi kis programmal elvégezhetjük a tár átszerveését:

```

0 REM **** P14. ****
1 :
2 :
100 FOR I=40960 TO 49151:REM BASIC-ROM MASOLAS
110 POKE I,PEEK(I):NEXT
120 FOR I=14*4096 TO 65535:REM OPERACIOS RENDSZER MASOLAS
130 POKE I,PEEK(I):NEXT
140 POKE 1,53

```

A 100-tól 130-ig terjedő sorok az operációs rendszert és a BASIC-et másolják át a ROM-ból az alatta lévő RAM-ba. A tár átrendezése után a 140. sorban a ROM-ot átkapcsoljuk RAM-ra. Így az operációs rendszer a továbbiakban a RAM-ban fut. A RAM területen már tetszőlegesen megváltoztatjuk az egyes billentyűk kódjait.

A német és az amerikai billentyűzet közötti eltérés az Y és Z billentyűk helyzetében van. A német billentyűzet előállításához ezt a két billentyűt fel kell cserélni, úgy, hogy a Z a T mellé, az Y pedig az X mellé kerüljön.

A billentyűk áthelyezéséhez szükségünk van a négy kódtáblázat kezdőcímeire.

A kódok módosításához meg kell határoznunk a kiválasztott billentyűk számát a billentyűzethez rendelt mátrixból. A mátrix elemeinek számozása a bal felső saroktól számítva soronként haladva 0-tól 63-ig tart. Az Y billentyű száma tehát 25, a Z billentyűé pedig 12. A módosítást a két billentyűszám felcserélésével végezhetjük el

- | | |
|---------------------------|----------------|
| 1. táblázat: | \$EB81 (60289) |
| 2. táblázat: SHIFT-tel | \$EBC2 (60354) |
| 3. táblázat: Commodore-ra | \$ECO3 (60419) |
| 4. táblázat: CTRL-lel | \$EC78 (60536) |

```

0 REM **** P15. ****
1 :
2 :
150 T1=60289:REM AZ ELSO TABLAZAT ALAPALLAPOT
160 T2=60354:REM A MASODIK TABLAZAT SHIFT BILLYENTYUVEL
170 T3=60419:REM A HARMADIK TABLAZAT COMMODORE BILLYENTYUVEL
180 T4=60536:REM A NEGYEDIK TABLAZAT CONTROL BILLYENTYUVEL
200 POKE T1+25,ASC("Z")
210 POKE T1+12,ASC("Y")

```

Ha a program végrehajtása után leütjük a Z billentyűt, Y jelenik meg a képernyőn, és megfordítva. A kisbetűk felcseréléséhez meg kell változtatnunk a második táblázat bejegyzéseit is:

```

0 REM **** P16. ****
1 :
2 :
220 POKE T2+25,ASC("Z")
230 POKE T2+12,ASC("Y")

```

A CTRL billentyű szerepét is megváltoztathatjuk:

```

0 REM **** P17. ****
1 :
2 :
240 POKE T4+25,ASC("Z")-64
250 POKE T4+12,ASC("Y")-64

```

A négy táblázat segítségével tehát $4 \cdot 64 = 256$ különböző karaktert határozhatunk meg. A RESTORE billentyű szerepét nem módosíthatjuk, mivel az közvetlenül kapcsolódik a processzor NMI-vezetékéhez. Az új billentyűzet-kiosztás a STOP/RESTORE lenyomásáig érvényben marad. A STOP/RESTORE leütésekor a rendszer ismét visszakapcsol a ROM-ra (alapállapotra), de ez is megakadályozható, ha a tárkiosztásra vonatkozó értéket a RAM-ban megváltoztatjuk. Illesszük be a fenti programba az alábbi sort:

190 POKE 64982, 53

Kicsit kényelmetlen az ismertetett eljárásban, hogy a billentyű számát esetlegesen a mátrix alapján ki kell számítani. Azonban ezt a feladatot is megoldhatjuk a gépben. A billentyűszám automatikus meghatározására szolgál az alábbi program:

```
0 REM **** P18. ****
1 :
2 :
100 DIM T(4):FOR I=1 TO 4:READ T(I):NEXT I
110 DATA 60289,60354,60419,60536
120 FOR I=14*4096 TO 65535:POKE I,PEEK(I):NEXT I
130 FOR I=40960 TO 49151:POKE I,PEEK(I):NEXT I
140 POKE 1,53:POKE 64982,53
1000 PRINT "KEREM NYOMJA MEG A ":PRINT "MÓDOSÍTANDÓ BILLENTYŰT":
1010 GET A$:IF A$="" THEN 1010
1020 PRINT A$
1030 A=ASC(A$)
1040 FOR J=1 TO 4: T=T(J)
1050 FOR I=0 TO 63:IF PEEK(T+I)<>A THEN NEXT:NEXT I
1060 PRINT "MELYIK BILLENTYŰRE CSERÉLI?":
1070 GET A$:IF A$="" THEN 1070
1080 PRINT A$
1090 POKE T+I,ASC(A$):GOTO 1000
```

3. FEJEZET

KÉNYELMES ADATBEVITEL

3.1 A kurzor pozicionálása és a kurzorpozíció lekérdezése

Az esztétikus képernyőtervek megvalósítása során a programozásban nagy segítséget jelentene, ha az input-output műveleteknél a kurzort a képernyő tetszőleges pozíciójára állíthatnánk. A Commodore 64-es alapszoftvere azonban a kurzor vezérlésére csak két utasítást szolgáltat: a TAB utasítást, amellyel a soron belül pozicionálhatunk, ill. a POS utasítást, amellyel lekérdezhetjük a kurzor aktuális oszloppozícióját.

Ugyanakkor az operációs rendszerbe be vannak építve az általános pozicionálási utasítás-hoz szükséges rutinok és rendszerváltozók. A kurzor aktuális helyzetének sor- és oszlop-koordinátája a tár két fix címén van, ezeket PEEK utasítással bármikor lekérdezhetjük.

```
0 REM **** P19. ****
1 :
2 :
100 PRINT "A KURZOR A"PEEK(214)". SORBAN":PRINT "ES A"PEEK(211)".OSZLOPBAN ALL."
READY.
```

Ha a kurzort a képernyő rögzített helyén akarjuk megjeleníteni, nem elég a sor-, ill. oszlop-koordinátát a 211-es és 214-es tárcímekre betölteni. A koordináták alapján kell kiszámítani az adott pozícióhoz tartozó képernyőtáron, ill. szintáron **belüli** tárcímeket. Az operációs rendszerben az S8640-es kezdőcímen levő rutin elvégzi ezt a **feladatot**.

Az alábbi BASIC programmal a kurzort a képernyő tetszőleges helyére állíthatjuk.

```
0 REM **** P20. ****
1 :
2 :
100 REM A KURZOR POZICIONALASA
110 INPUT "SOR SZAMA";Z
120 INPUT "OSZLOP SZAMA";S
130 POKE 214,Z
140 POKE 211,S
150 SYS 58640
160 PRINT "SZOVEG";
```

A képernyőszerkesztést rendkívüli módon megkönnyíti a bemutatott eljárás. Ha például olyan programot készítünk, amely az input-output műveletek végrehajtása közben üzeneteket küld a gépkezelőnek, gondot okozhat az üzenet szövegének elhelyezése, majd az aktuális művelet folytatása. Az ismertetett rutinok felhasználásával az üzenet kijelzése előtt lekérdezhetjük a kurzor aktuális pozícióját, elmozgathatjuk a kurzort az üzenet közlésére szánt rögzített sorra, majd kiírás után azon a képernyőpozíción folytathatjuk a következő műveleteket, ahol azt előbb abbahagytuk. Az elmondottakat az alábbi programrészlet szemlélteti:

```

0 REM **** P21. ****
1 :
2 :
300 Z=PEEK(214):REM SOR
310 S=PEEK(211):REM OSZLOP
320 POKE 214,0:REM A KURZOR A 0. SORBAN
330 POKE 211,3:REM A KURZOR A 3. OSZLOPBAN
335 SYS 58640
340 PRINT"KEREM TEGYE A LEMEZT A MEGHAJTOBA"
350 POKE 214,Z:REM A SOR VISSZAALLITASA
360 POKE 211,S:REM AZ OSZLOP VISSZAALLITASA
370 SYS 58640

```

A képernyőn a sorok számozása 0-tól 24-ig, az oszlopok számozása pedig 0-tól 39-ig terjed.

3.2 A kurzor ki- és bekapcsolása

A Commodore 64-es az INPUT utasítás végrehajtásakor, ill. parancsüzemmódban az aktuális képernyőpozíciót a kurzor villogtatásával jelzi. Az INPUT utasítással szemben a GET végrehajtásakor nem jelenik meg a képernyőn a villogó kurzor. Előfordulhat, hogy az adatbevitelt GET utasítással akarjuk elvégezni, de szükségünk lenne arra, hogy a kurzor most is villogjon, jelezve a gépkezelőnek, hogy hol tart a begépeléssel.

Az operációs rendszer a 204-es tárcím tartalma alapján végzi a kurzor ki-, bekapcsolását. Ha a tárcím tartalma nulla, a rendszer a kurzort villogtatja, ha pedig egy (vagy más nullától különböző érték), akkor a kurzort kikapcsolja.

A fentiekből következik, hogy a kurzor villogtatását a programunkból egyetlen POKE utasítással vezérelhetjük:

```
0 REM **** P22. ****
1 :
2 :
100 POKE 204,0:REM A KURZOR BEKAPCSOLASA
110 GET A$:IF A$="" THEN 110 :REM VÁRAKOZÁS EGY BILLENTYURE
120 POKE 204,1:REM A KURZOR KIKAPCSOLASA
130 PRINT A$;
```

Lehetséges, hogy a kurzor kikapcsolásakor a villogás abban a pillanatban szűnik meg, amikor a képernyőn látható a kis négyzet, és ez elrontja a képet. Minthogy a nulláslapon a 207-es tárcím tartalma mutatja, hogy a négyzet az adott pillanatban látszik vagy sem, ezt a problémát könnyen megoldhatjuk úgy, hogy lekérdezzük a 207-es cím tartalmát, a kurzort éppen abban a pillanatban kapcsoljuk ki, amikor a négyzet nem látható. Egészítsük ki a fenti programot az alábbi sorral:

```
115 IF PEEK (207) THEN 115:REM VÁRAKOZÁS A KURZOR KIKAPCSOLÁSÁRA
```

Az eljárás további alkalmazási lehetőségével találkozunk majd a 3.5 fejezetben.

3.3 Az összes billentyű automatikus ismétlése (repeat)

A Commodore 64-es használata során bizonyára észrevették, hogy a szóközbillentyű és a kurzor vezérlő billentyűk ismétléssel működnek, azaz amíg a billentyűt lenyomva tartjuk, automatikusan ismételnék. Az automatikus ismétlés a programok szerkesztése során különösen hasznos, és egy POKE utasítással az összes billentyűre nézve érvényesíthető.

- Ha a 650-es tárcím tartalma nulla, akkor csak a kurzor billentyűk ismételnék.
- Ha POKE utasítással a 650-es tárcímre 128-at írunk be, az összes billentyű ismételni fog.
- Ha az automatikus ismétlést teljesen ki akarjuk iktatni, a 650-es címre 64-et kell írunk.

```
0 REM **** P23. ****
1 :
2 :
100 POKE 655,128:REM MINDEN BILLENTYU ISMETEL
200 POKE 650,0 :REM CSAK A KURZOR ISMETEL
300 POKE 650,64:REM AZ ISMETLES KIKAPCSOLASA
```

Az ismétlés, ill. a késleltetés sebességét a 651-es és a 652-es tárcímek tartalma határozza meg. Ezeket a paramétereket csak az operációs rendszer áthelyezésével tudjuk tartósan megváltoztatni, mivel értéküket a rendszer indításkor mindig újra beállítja (l. a 2.6 és a 4.2 alfejezeteket).

3.4 A WAIT utasítás: Várakozás egy billentyű megnyomására

A WAIT rendkívül ritkán alkalmazott BASIC utasítás. Hogyan működik ez az utasítás?

WAIT A,B

Jelentése: beolvassa az A rekesz tartalmát (ugyanúgy, mint a PEEK utasítás), és végrehajt egy AND logikai műveletet az A és B tartalma között. Ha az eredmény nulla, az egész folyamat még egyszer megismétlődik, ha nem, a program fut tovább. Az utasítás működése alapján az A tárcímnek input/output portnak vagy egy külső egység címének, vagy egy olyan címnek kell lennie, amelynek értékét megszakító rutin változtatja. Ellenkező esetben ugyanis az utasítás vagy egyáltalán nem, vagy a végtelenségig várakozna.

A legérdekesebb felhasználás minden bizonnyal az egy meghatározott billentyű lenyomására való várakozás. A 653-as rekesz tartalma pl. arra utal, hogy lenyomott állapotban van-e a három speciális billentyű, a SHIFT, a CTRL vagy a COMMODORE billentyű. A WAIT utasítással mindaddig várhatunk, amíg a három billentyű valamelyikét le nem nyomták. Pl.:

```
0 REM **** P24. ****
1 :
2 :
100 PRINT "KEREM NYOMJA MEG A CTRL BILLENTYUT!"
110 WAIT 653,4:REM VARAKOZAS A CTRL-RE
```

A 110. sorban a program várja a CTRL billentyű lenyomását. A SHIFT és a COMMODORE billentyűre az alábbi WAIT utasításokkal várhatunk:

WAIT 653,1: VARAKOZAS A SHIFT BILLENTYURE

WAIT 653,2: REM VARAKOZAS A COMMODORE BILLENTYURE

Ha tetszőleges billentyű lenyomására akarunk várni, lekérdezhetjük a 203-as tárcímét. Ha egyetlen billentyűt lenyomva, a cím 64-et, egyébként a lenyomott billentyű mátrixszámát tartalmazza (l. a 2.3 fejezetet). A WAIT 203,64 utasítás mindaddig felfüggeszti a program futását, amíg egyáltalán van lenyomott billentyű.

A WAIT 203,63 utasítás csak egy billentyű lenyomása után folytatja a program végrehajtását. Az utóbbival helyettesíthetjük a GET utasítást, amelynél várakozás közben mindig le kell kérdezni az üres füzért.

```
0 REM **** P25. ****
1 :
2 :
100 WAIT 653,63
110 GET A$:PRINT A$;
```

A fenti WAIT utasítás csak akkor ér véget, amikor egy billentyűt lenyomunk. Ha már vannak adatok a billentyűzet-pufferben, a lenyomott billentyűk száma is alapul szolgálhat a WAIT lekérdezéséhez.

```
0 REM **** P26. ****
1 :
2 :
100 WAIT 198,255:REM VAN ADAT A KLAVIATURA PUFFERBEN?
110 GET A$:PRINT A$;
120 GOTO 100
```


3.5 Egy kényelmes INPUT rutin

Gyakran előfordul, hogy hibás adatbevitel miatt a program futása megszakad. A hiba forrása általában az, hogy INPUT utasítás után numerikus változót írtunk, de a végrehajtáskor nem numerikus karaktert (karaktereket) gépellünk be.

Ezt a hibát elkerülhetjük, ha INPUT utasítás helyett GET utasítást használunk. Ekkor viszont sok kényelmetlenséget okoz, ha nem egyetlen karaktert kívánunk beolvasni – hiszen a változót karakterenként kell összerakni. Másik hátránya, hogy a képernyőn nem jelenik meg a kurzor.

Ha el is fogadjuk a beviteli utasítások ismert hátrányait, és feltételezzük, hogy a gépkezelő nem követ el olyan hibát, amely közvetlenül az adatbevételnél okoz programmegszakítást, az adatbevételnél történt hibák a későbbi feldolgozásnál komoly gondot okozhatnak.

Az INPUT A utasítást például helyettesíthetjük az INPUT A\$ utasítással, elkerülve a programmegszakítást. Ez a megoldás numerikus adatok bevitelére is alkalmas, hiszen a VAL függvénnyel a karakteres változót numerikussá alakíthatjuk. A hiba lehetősége mégis fennáll, hiszen ha a gépkezelő pl. az 123R56 karaktersorozatot üti be, amikor a programban numerikus adatot várunk, az A=VAL(A\$) eredménye 123 lesz, ami feltehetően nem helyes.

Érvelhetünk azzal is, hogy az ilyen adatbeviteli hibák nem általánosak, és a „hobby programozók” programjaiból eredő hibás számításoknak nincsenek messzeható következményei. Véleményünk szerint azonban mindig célszerű megbízható programok megírására törekednünk, nehogy a továbbiak során bármikor is zavarba jöhessünk. A programok elkészítése közben nem szabad a saját szükségleteinkből kiindulni, mindig szem előtt kell tartani a megbízó igényeit.

Az alábbiakban egy olyan INPUT rutint mutatunk be az Olvasónak, amelyet bárki felhasználhat, és amely az említett programmegszakítási hibákat messzemenően kiszűri.

Érdemes a programot nagyon alaposan áttanulmányozni, hogy minél rugalmasabban alkalmazhassuk speciális igényű programozási feladatokban is. Először ismertetjük a programban használt változók rendeltetését. Az alábbi változók értékét a rutin meghívása előtt a feladat jellegétől függően a használatnak kell meghatározni:

| | |
|------|---|
| MN=0 | Numerikus adatbevitel |
| MN=1 | Alfanumerikus adatbevitel |
| ML=0 | Az IL-ben megadott hosszúság kötelező! |
| ML=1 | Az IL-ben a maximális hosszúságot kell megadni! |
| IL | Az adat kötelező vagy maximális hosszúsága. |

A rutin további változói:

| | |
|------|--|
| CC | Érvényes karakterek száma az IN\$-ban. |
| CS | A kurzor pillanatnyi oszloppozíciója. |
| CZ | A kurzor pillanatnyi sorpozíciója. |
| CP | Az inputmező INSERT által előállított hosszúsága. |
| MS | A kurzor legmagasabb oszloppozíciója az adatbevitel során. |
| G\$ | Az utoljára végrehajtott GET-tel beolvasott karakter |
| IN\$ | A beolvasott adat. |

A rutinban használt tárcímek:

| | |
|---------|--|
| 204 = 0 | A kurzor bekapcsolása. |
| 204 = 1 | A kurzor kikapcsolása. |
| 205 | A villogás frekvenciájának számlálója. |
| 207 = 0 | A kurzor a KI fázisban. |
| 207 = 1 | A kurzor a BE fázisban. |
| 211 | Az oszlop pozíciója. |
| 214 | A sor pozíciója. |

A rutin meghívása előtt meg kell nyitni a képernyőt az OPEN 1,3 utasítással, mivel a 35680-as sorban GET # 1 utasítással olvassuk be az adatokat

A rutin lépései:

| | |
|-------------|--|
| 35020 | A szükséges változók kezdőértékének meghatározása és a kurzorpozíció tárolása a 35680. sor GET # 1 utasításához. |
| 35060 | Egy karakter beolvasása a klaviatúráról. |
| 35080 | Ha a beolvasott karakter a RETURN volt, az adatbevitelnek vége, az adat hosszúságát az ML értékétől függően ellenőrizni kell. |
| 35100–35130 | DELETE billentyű használata után a hosszúságot és a pozíció-számlálót aktualizálni kell attól függően, hogy az input mező csak érvényes karaktereket tartalmaz (CP = 0), vagy az INSERT-tel bővült (CP < > 0). |
| 35140 | Az INSERT-et csak akkor szabad végrehajtani, ha ezzel nem léptük túl az IL-ben megadott hosszúságot. |
| 35160–35180 | A kurzor jobbra és balra mozgatása közben gondoskodni kell arról, hogy ne hagyjuk el az input mezőt. |
| 35200 | Ugrás az adatvizsgálatra, MN értékétől függően. |
| 35220–35240 | Numerikus adatbevitelnél, ha a kurzor az adatmezőn belül van, a program elfogadja a karaktert. Az érvényességi tartományt a 47 és az 58 kódokkal szabtuk meg. Az általunk megadott tartomány numerikus adatbevitelnél csak 0-tól 9-ig terjed. Az érvényességi tartományt tetszés szerint bárki megszabhatja a C-64-es felhasználói kézikönyvben található kódtáblázat alapján. |
| 5300–35380 | Alfanumerikus ellenőrzés. |
| 35400 | Az adatbevitel nem érhet véget, ha az adat hossza még kisebb az előírt hosszúságnál (ML = 0). |
| 35600–35690 | Az IL-ben megadott hosszúság eléréseig a beütött karaktereket gyűjtjük az INS\$ változóban. Előzetesen a kurzort a kezdőpozícióra állítjuk, hogy a GET # 1 utasítást a mező elejétől indíthassuk. |
| 36000–36060 | A kurzort kikapcsoljuk és a GS-ban lévő karakter megjelenik a képernyőn. |

A rutin sorszámozása természetesen tetszőleges. Lényegesen megkönnyíti azonban a programozást egy olyan segédprogramokat tartalmazó könyvtár kialakítása, amelyben az egyes rutinokat azonos sorszámozással tároljuk.

Az alábbiakban közöljük az INPUT rutin listáját.

```

0 REM **** P27. ****
1 :
2 :
35000 REM ADATBEVITEL BILLENTYUZETROL
35020 IN$="":CC=0:CS=PEEK(211):CZ=PEEK(214):CP=0:MS=0
35040 POKE204,0:REM KURZOR BE
35060 GET G$:IF G$="" THEN 35060
35080 G=ASC(G$):IF G=13 THEN ON ML GOTO 35400,35600
35100 IF G=20 AND CP>0 THEN CP=CP-1:GOSUB 36000:GOTO 35060:REM TORLES
35120 IF G=20ANDCC=0ANDPEEK(211)>CSTHENCC=CC-1:MS=MS-1:CP=CP-1:GOSUB36000:GOTO350
60
35130 IF G=20 AND PEEK(211)>CS THENMS=MS-1:GOSUB36000:GOTO35060
35140 IF G=14ANDCP+MS<ILTHENCP=CP+1:MS=MS+1:GOSUB36000:GOTO35060:REM BESZURAS
35160 IF G=29ANDPEEK(211)<=CS+IL-1THENGOSUB36000:GOTO35060:REM KURZOR JOBBRA
35180 IF G=157ANDPEEK(211)>CSTHENGOSUB36000:GOTO35060:REM KURZOR BALRA
35200 ON MN GOTO 35300
35220 IF G>47ANDCC<ILANDPEEK(211)<=CS+IL-1THENCC=CC+1:GOSUB36000:GOTO35360
35230 GOTO35360
35240 IF G>47ANDPEEK(211)<=CS+ILTHENGOSUB36000:GOTO35360
35300 IF G<48 OR (G>57 AND G<65) OR (G>90ANDG<193) OR G>218THEN35060
35320 IFCC<ILANDPEEK(211)<=CS+IL-1THENCC=CC+1:GOSUB36000:GOTO35360
35340 IFPEEK(211)<CS+ILTHENGOSUB36000
35360 IFCP>0THENCP=CP-1
35380 GOTO35060
35400 IFCC<>ILTHEN35060
35600 POKE205,2
35620 IFPEEK(207)<>0THEN35620
35640 POKE204,1
35660 POKE211,CS:POKE214,CZ
35670 IFCC=0THENRETURN
35680 GET#1,G$:IFG$=CHR$(13) THENIN$=LEFT$(IN$+"",IL):RETURN
35682 IN$=IN$+G$
35684 IF LEN(IN$)<ILTHEN35680
35690 RETURN
36000 POKE205,2
36020 IFPEEK(207)<>0THEN36020
36040 PRINTG$;:IFPEEK(211)>MSTHENMS=PEEK(211)
36060 RETURN

```

Hogyan használjuk a bemutatott programot?

Tegyük fel, hogy egy raktárnyilvántartási rendszer adatbeviteli programjában az anyagok jellemzőit kell beolvasnunk a billentyűzetről (cikkszám, anyagnév, egységár, mennyiség stb.). Minden cikkszám hatjegyű numerikus adat. A cikkszámot beolvasó programrész a rutin felhasználásával:

```

0 REM **** P28. ****
1 :
2 :
100 IL=6:MN=0:ML=0
110 PRINT "CIKKSZAM";:GOSUB 35000
120 IN=VAL(IN$)

```

A beolvasott cikkszám a programrész lefutása után az IN változóban áll rendelkezésre, és egészen biztosan hat numerikus karakterből áll.

A cikkszám beolvasásához hasonló az anyagnév beolvasása azzal a különbséggel, hogy az anyagnév alfanumerikus adat és általában nem fix hosszúságú. Ha relatív file-szervezéssel dolgozunk, amelyben az adatmezők és a rekordok hossza rögzített, meg kell adnunk az anyagnév lehetséges maximális hosszát. Legyen a maximális hossz esetünkben 10 karakter, amelyet nem kötelező kitölteni:

```

0 REM **** P29. ****
1 :
2 :
100 IL=10:MN=1:ML=1
110 PRINT "MEGNEVEZES";:GOSUB 35000

```

Az IN\$-ban visszakapott anyagnevet szükség esetén üres karakterekkel kiegészíthetjük 10 karakter hosszúságúra.

Az egységár általában változó hosszúságú adat, és természetesen csak numerikus karaktereket tartalmazhat. Tegyük fel, hogy maximálisan 8 mezőszélességű számadatokkal dolgozunk:

```

0 REM **** P30. ****
1 :
2 :
100 IL=8:MN=0:ML=1
110 PRINT "EGYSEGAR";:GOSUB 35000
120 IN=VAL(IN$)

```

A beolvasott egységárat ismét az IN változóban kapjuk vissza, és folytathatjuk a programot a mennyiség beolvasásával. Reméljük, hogy ez a rutin hasznos szolgálatot tesz az Olvasónak, megkíméli a hibás adatbeviteltől eredő bosszúságoktól, és a programban bemutatott trükkök (a kurzor pozicionálása, az újszerű adatbevitel) mindenkit a könyv további tanulmányozására csábít.

3.6 A CBM 64-es „egere”

A személyi számítógépesek táborában megjelent egy új fogalom: az egér.

Milyen számítástechnikai eszköz rejtőzik e fedőnév mögött? A videojátékok ismerői már biztosan találkoztak az ún. pörgő golyóval, ami a joystickhoz hasonló, mozgást vezérlő berendezés. Újdonság, hogy míg a joystickkal a képernyőn megjelenő figurát csak nyolc lehetséges irányba, a pörgő golyóval a sík tetszőleges irányába elmozdíthatjuk. Az elmozdulás irányát két, (az X-, ill. az Y-tengelyre vonatkozó) ún. szögmódoló érzékeli, és továbbítja a számítógéphez.

Az egér is a pörgő labdához hasonlóan, minden irányban szabadon mozgatható. A tokozott egeret az asztalon húzogatjuk. A surlódástól a golyó forogni kezd, miközben a két szögmódoló pillanatnyi állását a beépített interface továbbítja a számítógéphez. Megismerve a CBM 64-es gép „háziállatának” képességeit, joggal merülhet fel a kérdés, hogy ez a tulajdonképpeni játékszer milyen szerepet tölthet be egy komoly számítástechnikai feladat megoldásában.

A széles használati kör számára készített programnál vagy programrendszernél alapvető követelmény, hogy az áttekinthető és könnyen kezelhető legyen. Ezt a célt szolgálja az ún. menütechnika. A képernyőn megjelenő menü tartalmazza mindazokat a lehetőségeket, amelyek közül a használó a feldolgozás során választhat. A választást általában kétféleképpen végezhetjük: a menüpont azonosító kódját leütjük; vagy a képernyőn a kurzort a menüpontot azonosító szövegmezőre mozgatjuk.

Ha megkérdezzük az ergonómiában jártas szakembert, ő a különféle megoldások közül biztosan elvetné a billentyűzeten való keresgélést.

Legcélszerűbb a menütechnikát és az egér szolgáltatta mozgatási lehetőséget párosítani, így a gépkezelő egy karosszékből kényelmesen elhelyezkedve, esztétikusan megszerkesztett képernyőn a figura (vagy kurzor) mozgását a vele egyirányú mozgást végző egérrel irányíthatja. A kívánt mezőre érve, választását a tűzgomb megnyomásával jelezheti.

Feltehetően nem rendelkezik minden Olvasó CBM 64-es gépéhez egérrel, ezért egy olyan programmal illusztráljuk a fentieket, ami elvben a leírtak szerint működik, de a mozgatót nem egérrel, hanem a kettes portra csatlakoztatott joystickkal oldja meg.

Tekintsük először a programban használt változók és tárcímek feladatait:

Változók:

| | |
|------|---|
| RO\$ | Az RVS üzemmód bekapcsolásának karaktere |
| RF\$ | Az RVS üzemmód kikapcsolásának karaktere |
| A\$ | A mindenkori karakter |
| B\$ | A RETURN megnyomása után az előtte leütött karaktert tartalmazza |
| A | Négy sort és négy oszlopot tartalmazó indexes változó, a képernyő adott sorában és adott oszlopában kijelzett karakter tárolására |
| DR | Az 56322-es adatirány-regiszter eredeti tartalma. (A program lefutása után az eredeti értéket ugyanis vissza kell állítani.) |
| J | A joystickról a kettes porton leolvasott érték |
| JS | A joystick oszloppozíciója |
| JZ | A joystick sorpozíciója |
| PS | A PRINT utasítás oszloppozíciója |
| PZ | A PRINT utasítás sorpozíciója |

| | |
|---|--|
| S | A joystick helyzetének oszlopa az A változó első indexének meghatározására [A(x,y)]. |
| Z | Ua. mint az előbbi, sorra vonatkoztatva. |

Az érintett tárcímek:

| | |
|-------|---|
| 56322 | A kettes port adatirány-regisztere |
| 58643 | A kurzorpozíciót lekérdező rendszerrutin |
| 58636 | A kurzort pozicionáló rendszerrutin |
| 781 | Az X indexregiszter tartalma, melyet SYS utasítással betöltünk vagy felülírunk. |
| 782 | Ua. mint előbb az Y indexregiszterre vonatkozóan |
| 204 | Ha értéke nulla, a kurzor be van kapcsolva, ha 1, ki van kapcsolva |
| 205 | A kurzor villogásának frekvenciája |
| 207 | Ha értéke nulla, a kurzor nem látható (KI fázis), ha nem nulla, látható (BE fázis). |

A következőkben lépésről lépésre elemezzük a program működését:

| | |
|-----------|--|
| 1 | Mivel a billentyűzet és a vezérlő portok a CBM 64-esen azonos perifériavonalat használnak, a billentyűzetet kikapcsoljuk. A program lefutása után az 56322 tárcímre (adatirány-regiszter) visszatöltjük a DR változóban megőrzött eredeti tartalmat, egyébként ugyanis a gép nem érzékelné a billentyűzetet. A mintaprogramban csak a RUN/STOP billentyű él. |
| 10–50 | A menü felépítése a képernyőn |
| 60–560 | Az indexes változó feltöltése az 1–4. képernyősorban kiírt karakterek ASCII kódjával. A változóban tárolt kód alapján az első négy sor bármely pozícióján álló karaktert visszanyerhetjük. |
| 680 | Az 5000-es sorban található szubrutinban kapott karakter kiírása a képernyőre. |
| 700 | A PRINT utasítást követő kurzorpozíciót megőrizzük a menükép későbbi visszaállításához (720-as sor). Egyébként a teljesen kitöltött képernyő sorai új kiíratásnál feltölödne. |
| 760 | Az új mező felépítése közben megváltozott kurzorpozíciót visszaállítjuk. |
| 780 | Ha az éppen leütött karakter a RETURN, az adatbevitel véget ér. A beolvasott adatot a B\$-ban tároljuk. |
| 5020–5140 | A kurzorpozíció tárolása után a kurzort kikapcsoljuk, majd a nulladik pozícióra állítva ismét bekapcsoljuk. |
| 5160 | A botkormány aktuális értéke a kettes portról a J változóba kerül. |
| 5170 | Késleltető ciklus a kurzor ellenőrzéséhez. |
| 5180–5340 | A kurzor mozgása követi a joystick mozgását. A tűzgomb megnyomása után (5260) a kurzor alatti karakter bekerül az A\$ változóba (6010) |
| 6010–6160 | A kurzorpozíció alapján kiszámítjuk az A változó indexeit, majd az ACII kódból a megfelelő karaktert (A\$) (6060) A C-64-es duplasor-szervezésű, így az oszlopszámláló maximális értéke 80, annak ellenére, hogy a képernyősor maximum 40 karaktert tartalmazhat. A 6050-es sorban korrigáljuk az A változó oszlopindexét. |

A program kezelése rendkívül egyszerű. A RUN parancs után a képernyő bal felső sarkában megjelenik a kurzor, amit a kettes portra csatlakoztatott joystickkal tudunk mozgatni. Ha a kurzor azon a karakteren áll, amelyre szükségünk van, nyomjuk meg a tűzgombot. A karakter

bekerül a B\$ változóba. A helyes választást ellenőrizhetjük, hiszen a választott karakter néhány sorral lejjebb megjelenik a képernyőn. Mindaddig, míg a RETURN-t le nem ütjük, a program gyűjti a kiválasztott karaktereket a B\$ változóban. Az adatbevitel végét jelző RETURN után egyrészt a B\$ változó tartalmát tetszőleges célra felhasználhatjuk, másrészt folytathatjuk az adatbevitelt a következő adatsor összeállításával.

A program listája:

```
0 REM **** P31. ****
1 :
2 :
4 DR=PEEK(56322):POKE 56322,224:R0$=CHR$(18):RF$=CHR$(146)
5 PRINT CHR$(147):GOSUB 10:GOTO 60
10 PRINT CHR$(19)" , - . / 0 1 2 3 4 5 6 7 8 9 " ;
20 PRINT " 5 A B C D E F G H I J K L " ;
30 PRINT " M N O P Q R S T U V W X Y Z " ;
40 PRINT " "R0$"RET"RF$" "R0$"DEL"RF$" "R0$"F1"RF$" "R0$"F3"RF$" "R0$"F5"RF$";
45 PRINT " "R0$"F7"RF$" "
50 RETURN
60 DIM A(4,40)
100 FOR I=0 TO 13
120 A(0,I*2+1)=I+44
140 NEXT I
180 FOR I=0 TO 12
200 A(1,I*2+2)=I+64
220 NEXT I
260 FOR I=0 TO 13
280 A(2,I*2+1)=I+77
300 NEXT I
340 FOR I=0 TO 3
360 A(3,I)=13
380 NEXT I
420 FOR I=5 TO 7
440 A(3,I)=20
460 NEXT I
500 FOR I=0 TO 3
520 A(3,I*2+9)=I+133
560 NEXT I
580 PRINT:PRINT
600 B$="":X=FRE(0)
640 GOSUB 5000:REM
680 PRINT A$;
700 SYS 58643:PZ=PEEK(211):PS=PEEK(214)
720 GOSUB 10
760 POKE 211,PZ:POKE 214,PS:REM SYS 58636
780 IF ASC(A$)=13 THEN 600
800 B$=B$+A$
820 GOTO 640
5000 REM
5001 REM ***** JOYSTICK
5002 REM
5020 SYS 58643:REM
5060 PZ=PEEK(781):PS=PEEK(782)
5070 POKE 205,3
5080 IF PEEK(207) THEN 5080
5090 POKE 204,1
5100 POKE 781,0:POKE 782,0:JZ=0:JS=0
5120 SYS 58636:REM JOYSTICK KURZOR BEALLITASA
5140 POKE 204,0:REM A KURZOR BEKAPCSOLASA
5160 J=PEEK(56320):REM A JOYSTICK LEKERDEZESE
5170 FOR I=0 TO 50:NEXT I
5180 IF (JAND1)=0 THEN JZ=JZ-1
5200 IF (JAND2)=0 THEN JZ=JZ+1
5220 IF (JAND4)=0 THEN JS=JS-1
5240 IF (JAND8)=0 THEN JS=JS+1
```

```

5260 IF (JAND16)=0 THEN 6000
5280 IF JZ<0 THEN JZ=0
5281 IF JS<0 THEN JS=0
5282 IF JS>30 THEN JS=30
5283 IF JZ>3 THEN JZ=3
5285 POKE 205,3
5290 POKE 204,1
5300 POKE 781,JZ:POKE 782,JS:SYS 58636
5340 GOTO 5140
6000 REM
6001 REM ***** HOZZARENDELES A KARAKTERHEZ *****
6002 REM
6010 POKE 205,3
6015 IF PEEK(207) THEN 6015
6017 POKE 204,1
6020 SYS 58643:REM KURZORPOZICIO BETOLTESE
6040 Z=PEEK(781):S=PEEK(782)
6050 IF S>39 THEN S=S-40
6060 A$=CHR$(A(Z,S))
6100 POKE 781,PZ:POKE 782,PS
6120 SYS 58636:REM KIIRASI POZICIO BETOLTESE
6160 RETURN

```


4. FEJEZET

BASIC HALADÓKNAK

4.1 BASIC sor előállítás BASIC-ből

Megkísérelte már az Olvasó egy tökéletesen általános program megírását?

Az általános megjelölés alatt azt értjük, hogy a program tetszés szerinti változókkal és konstansokkal tetszés szerinti aritmetikai műveletet tud elvégezni. Az olvasó válasza feltehetően: nem, hiszen a program lényegét tekintve egy előre megadott algoritmus, amely a feldolgozás alatt nem változik.

Haladó programozók fejében azonban előbb utóbb felmerül egy saját szövegszerkesztő program megírásának gondolata. Ha a tárolt szövegben vannak numerikus mezők, azokkal többnyire aritmetikai műveletet kell végezni.

Egy kereskedő számláinak feldolgozása közben például összegezni kell az egyes számlákon szereplő értékeket, összegezés után szorzási művelettel ki kell számítani a többletérték-adót, majd a kapott számhoz hozzá kell adni a csomagolás és szállítás költségét.

A feladat algoritmusát nagyon könnyű áttekinteni, és úgyszintén könnyű a BASIC nyelvű programot elkészíteni. Ezt a programot azonban egy szőnyegkereskedő feltehetően már nem tudja használni, hiszen a számlaérték nála az eladott termék egységárától és a méterben megadott mennyiségtől függ. Világos, hogy a szőnyegkereskedő számára más képleteken alapuló, új programot kell készíteni.

Ha így folytatjuk a gondolkodást, kiderül, hogy ahányféle kereskedő van, annyiféle egymástól alig eltérő, de mégis más-más programot kell írni, ami nem gazdaságos. Sokkal praktikusabb lenne, egy olyan általános célú programot készíteni, amelynek feldolgozás közben meg lehet adni, hogy a tárolt adatok között mikor milyen aritmetikai műveletet végezzen el.

Eddigi ismereteink alapján ez lehetetlen, hiszen azokat az aritmetikai kifejezéseket, amelyekben egy-egy változó értéket kap, előzetesen BASIC utasításként be kell építeni a programba. Most bemutatunk egy olyan programot, amely képes arra, hogy futás közben beépítsen (saját magába) egy kívülről megadott BASIC sort. A program gépi kódú részt is tartalmaz, de ez ne okozzon különösebb gondot az Olvasónak, hiszen közöljük a BASIC betöltőprogramot is. Ismerkedjünk meg a programban használt változókkal:

| | |
|------|---|
| TM | A tár végcíme |
| VL | A változóterület kezdőcíme (alsó byte) |
| VH | A változóterület kezdőcíme (felső byte) |
| VT | A változóterület teljes kezdőcíme |
| BU | Az input pufferen belüli mutató |
| BC | A puffer feltöltését vezérlő változó |
| CA\$ | A belrt aritmetikai kifejezés |
| RE | Az eredmény a rutin lefutása után |

Érintett tárcímek:

- 45-46 A változóterület kezdete (mutató)
- 47-48 Az indexes változók kezdete (mutató)
- 49-50 A füzérváltozók kezdete (mutató)

- 56 A BASIC terület vége (HB)
 40448 Az előállított BASIC sor (50100)
 40704 Annak a rutinnak a kezdete, amely az input puffer tartalmából előállítja a BASIC sort és elhelyezi a 40448-as címtől kezdve.

A program dokumentációja:

- 1 A RAM-top-ot beállítjuk 40448-ra, előírt helyezkedik el ugyanis a gépi kódú rutin.
 2-6 A változókezdlet-mutatót átállítjuk, mivel az 50099-es kapcsolósor ide kerül.
 10-14 Előkészítjük az 50100 és 50110-es sorok tárbeli helyét. A tényleges sorokat majd a gépi kódú rutin készíti el, itt egyszerű PRINT és RETURN utasításokat írunk a helyükre.
 20-30 Az 50099-es kapcsolósort elhelyezzük közvetlenül a BASIC program mögé.
 32-50 Gépi kódú programrész
 60-70 A gépi kódú részt a 40704-es címtől kezdve elhelyezzük a tárban.
 50040 A BASIC kifejezés beolvasása a CA\$ változóba. Az adatbevitelnél csak a kifejezést kell megadni a teljes egyenlőség helyett, azaz a program az „RE= ” utáni részt kéri.
 50050 A CA\$ változó tartalmát áttöltjük a sorhoz tartozó input pufferbe (50075-ig).
 50080 Ugrás a gépi kódú részre, ahol a BASIC sor előállítás történik.
 50095 Az aritmetikai kifejezés kiértékelése, az eredmény az RE változóba kerül.

Az érdeklődőknek közöljük a gépi nyelvű programot is.

```

0 REM **** P32. ****
1 ;
2 ;
100 : 9F00 A5 7A LDA $7A ;A BASIC-MUTATO TARTOLASA
110 : 9F02 BD FF 9F STA $9FFF
120 : 9F05 A5 7B LDA $7B
130 : 9F07 BD FE 9F STA $9FFE
140 : 9F0A A5 14 LDA $14
150 : 9F0C BD FD 9F STA $9FFD
160 : 9F0F A5 15 LDA $15
170 : 9F11 BD FC 9F STA $9FFC
180 : 9F14 A9 0B LDA $0B ;AZ INPUTPUFFER KEZDETE
190 : 9F16 B5 7A STA $7A
200 : 9F18 20 79 A5 JSR $A579 ;A 'CRUNCH' RUTIN HIVASA
210 : 9F1B A2 00 LDX #0
220 : 9F1D BD 00 02 XX LDA $0200(X) ;A SOR ATVITELE A 40453 CIMRE
230 : 9F20 F0 06 BEQ YY ;UGRAS,HA VEGE
240 : 9F22 9D 05 9E STA $9E05(X)
250 : 9F25 E8 INX
260 : 9F26 D0 F5 BNE XX
270 : 9F28 A9 3A YY LDA $03A ;A SOR MOGE
280 : 9F2A 9D 05 9E STA $9E05(X)
290 : 9F2D A9 8E LDA $08E ;A 'RETURN' ELHELYEZESE
300 : 9F2F 9D 06 9E STA $9E06(X)
310 : 9F32 A9 00 LDA #0 ;A SOR VEGENEK TARTOLASA
320 : 9F34 9D 07 9E STA $9E07(X)
330 : 9F37 9D 08 9E STA $9E08(X)
340 : 9F3A 9D 09 9E STA $9E09(X)
350 : 9F3D AD FF 9F LDA $9FFF ;A BASIC-MUTATO VISSZATOLTESE
360 : 9F40 B5 7A STA $7A
370 : 9F42 AD FE 9F LDA $9FFE
380 : 9F45 B5 7B STA $7B
390 : 9F47 AD FD 9F LDA $9FFD
400 : 9F4A B5 14 STA $14
410 : 9F4C AD FC 9F LDA $9FFC
420 : 9F4F B5 15 STA $15
430 : 9F51 60 RET ;VISSZA A BASIC-RE

```

A következő (P.33) program két részből épül fel:

Az 1–70-es programsorokra csak a futtatás elején, egyszer ugrunk. Nagyon fontos, hogy ezek a sorok mindig elől legyenek, tehát nem szabad a programot átsorszámozni! Ellenkező esetben ugyanis az 1–6-os sorokban szereplő változók értékét elronthatjuk.

A második rész a CA\$-ban megadott képletből BASIC sort készít, kiértékeli a kifejezést és az eredményt a RE változóba helyezi.

Ha futtatás közben nem akarjuk megváltoztatni a kifejezésben használt műveleteket, az 50100-as sort átugorhatjuk. Ha az Olvasó a programot rutinként szeretné használni, ügyeljen arra, hogy a főprogram sorszámozása csak 49999-ig tarthat. A program utolsó sorainak számozását feltétlenül 50000-tól kell kezdeni.

A képlet meghatározásakor csak az egyenlőségjel utáni részt kell beírni, pl.: $75/2 * V1 - V2 + SQR(V3)$.

A kifejezés értékét az 50050-es sorban kapjuk meg.

FIGYELEM!

A programot az első futás után már nem szabad megváltoztatni! Ha erre mégis szükség van, ki kell adnunk egy NEW parancsot, majd ismét be kell tölteni a programot és betöltés után módosítani.

A programból előállított BASIC sor ugyanis nem a BASIC program alá, hanem a tár tetejére kerül. Az ilyen jellegű változtatás után egy BASIC sor törlését, vagy új sor beillesztését nem tudja hibamentesen elvégezni az operációs rendszer.

```
0 REM **** P33. ****
1 POKE 56,158:CLR
2 IF PEEK(45)+2>255 THEN POKE 45,2-(256-PEEK(45)):POKE 46,PEEK(46)+1:GOTO 6
4 POKE 45,PEEK(45)+2
6 POKE 47,PEEK(45):POKE 48,PEEK(46):POKE 49,PEEK(45):POKE 50,PEEK(46)
8 TM=40448
10 POKE TM,0:POKE TM+1,7:POKE TM+2,158:POKE TM+3,180:POKE TM+4,195
12 POKE TM+5,153:POKE TM+6,0:POKE TM+7,13:POKE TM+8,158:POKE TM+9,190
14 POKE TM+10,195:POKE TM+11,142:POKE TM+12,0:POKE TM+13,0:POKE TM+14,0
20 VL=PEEK(45):VH=PEEK(46):VT=VH*256+VL
30 POKE VT-4,1:POKE VT-3,158:POKE VT-2,179:POKE VT-1,195
32 DATA165,122,141,255,159,165,123,141,254,159,165,20,141,253,159,165,21
33 DATA141,252,159,169,11,133,122,32,121,165
34 DATA162,0,189,0,2,240,6,157,5,158,232,208,245,169,58,157,5,158
36 DATA 169,142,157,6,158,169,0,157,7,158,157,8,158,157,9,158
40 DATA 173,255,159,133,122,173,254,159
50 DATA 133,123,173,253,159,133,20,173,252,159,133,21,96
60 FOR I=40704 TO 40785
70 READ MC:POKE I,MC: NEXT I
50000 REM KALKULATOR *****
50040 BU=523:INPUT "KALK. ":CA$
50050 POKEBU,ASC("R"):POKEBU+1,ASC("E"):POKEBU+2,ASC("="):BU=BU+2
50060 FOR I=1 TO LEN(CA$)
50070 POKE BU+I,ASC(MID$(CA$,I,1)):NEXT I
50073 BC=LEN(CA$)+1
50075 POKE BU+BC,0:POKE BU+BC+1,0:POKE BU+BC+2,0
50080 SYS 40704
50095 GOSUB 50100
50097 RETURN
```

Végezetül ejtsünk néhány szót a program továbbfejlesztési lehetőségeiről.

Tegyük fel, hogy egy speciális számítástechnikai feladat programozástechnikai oldalról oda vezetett, hogy BASIC sort kellett beépítenünk egy programba futás közben. Az első lépés

megtételé további általánosításra ösztönöz: ki kellene dolgozni egy olyan módszert, amellyel nemcsak egy-egy sort, hanem ezek láncolatát is – beleértve a ciklusokat és ugrásokat is -- be lehet építeni a futó programba.

Az ismertetett eljárást az egymást követő BASIC sorok tárbeli elhelyezésével kell bővítenünk, ügyelve arra, hogy a gépi kódú rutin a beolvasott BASIC sort ne azonos tárterületre, hanem a sorok hosszától függően egymás mögé helyezze.

Gondoskodni kell arról is, hogy a folytatósor mutatóját (minden BASIC sor első két byte-ja) megfelelőképpen beállítsuk, ami egyetlen sor beillesztésénél nem okozott problémát.

Reméljük, sikerült megsejtenni, hogy ez a problémakör tulajdonképpen egy programgenerátor elkészítésének alapelveit érinti. Programgenerátor alatt egy olyan programot értünk, amely egy adott témakörben felhasználható adott programnyelvű (nem feltétlenül BASIC) programok önálló megszerkesztésére, azaz tulajdonképpen egy ún. makro-interpreter.

4.2 A BASIC interpreter átmásolása a RAM-ba

A Commodore 64-es előnye a többi Commodore gépekhez képest az a lehetőség, hogy a processzor teljes címtérülete (64 kbyte) áthelyezhető a RAM területre. A teljes operációs rendszert, a BASIC interpretert, átalakíthatjuk anélkül, hogy a ROM egyetlen alapelemét ki kellene cserélnünk. A módosított operációs rendszert, ill. interpretert át kell töltenünk a RAM-ba, majd közölni kell a géppel, hogy a ROM-ot kapcsolja ki, és helyette aktivizálja a megfelelő RAM területet. (Ez utóbbi egyetlen POKE utasítást igényel.)

Ha nem akarjuk a teljes BASIC interpretert kicserélni, hanem csak bizonyos részeit megváltoztatni – például egy saját függvényt definiálni, vagy meglévő függvényeket, utasításokat módosítani –, be kell tölteni a teljes interpretert a RAM-ba, ahol minden változtatást elvégeztünk, majd átkapcsolva RAM területre, dolgozhatunk a saját igényünk szerint kialakított interpreterrel.

Nézzük meg közelebbről a Commodore 64-es társzervezését!

Amikor a gépet bekapcsoljuk, automatikusan működésbe lép a ROM-beli operációs rendszer és a BASIC interpreter. PEEK utasítással olvasva erről a területről, a tényleges ROM tárcímek tartalmát kapjuk meg. Ezzel szemben, ha POKE utasítással akarunk írni a tárba, az mindig a RAM területre vonatkozik, függetlenül attól, hogy a ROM vagy a RAM van bekapcsolva. Következésképpen egy egyszerű ciklussal átmásolhatjuk a ROM-beli operációs rendszert az alatta fekvő RAM területre:

```
FOR I=A TO E : POKE I, PEEK(I) : NEXT
```

ahol A a tárterület kezdőcíme, E pedig a végcíme. A BASIC kezdő-, ill. végcíme: 40960(\$A000) és 49151(\$BFFF); az operációs rendszeré: 57344(\$E000) és 65535(\$FFFF).

Másolás után mindaddig továbbra is a BASIC-ROM működik, amíg a változást nem közöljük a géppel. A ROM/RAM átkapcsolást az 1-es tárcím tartalmának (processzor port) beállításával lehet megvalósítani. Alapállapotban az 1-es tárcím tartalma 55. Ha a BASIC-et a RAM-ban akarjuk futtatni, a POKE 1,54 utasítást kell végrehajtanunk.

Figyelem! Ezt a POKE utasítást csak a fenti ciklus (40960-tól 49151-ig) lefutása után szabad kiadni, ellenkező esetben a gép működésképtelenné válik! Ha az operációs rendszert is áthelyezzük a RAM-ba, a RAM-beli futtatás csak a BASIC-ROM-mal együtt lehetséges, ugyanígy ha átkapcsolunk erre a RAM területre, automatikusan bekapcsolódik az alatta lévő BASIC-ROM is (l. 2.6 fejezetet). Az átkapcsolást a POKE 1,53 utasítás eredményezi. A BASIC interpretert a következő lépésekben módosíthatjuk: átmásoljuk a ROM-ból a RAM-ba; végrehajtjuk a megfelelő változtatásokat; tároljuk azt a programot, amellyel a változtatást végeztük (hogy legyen módunk megismételni az eljárást, ha hibát követtünk el).

A biztonsági művelet után kapcsoljunk át a BASIC-RAM-ra!

4.3 A negatív számok kiküszöbölése az FRE függvény értékkészletéből

Megtörtént már az olvasóval, hogy bekapcsolás után a Commodore 64-es gépe 38911 szabad byte-ot jelzett, de a

```
PRINT FRE(0)
```

utasítás – 26627 szabad byte-ról számolt be?

Ahhoz, hogy a tényleges értéket megkapjuk, a kapott negatív számhoz hozzá kell adni 2^{16} -t, azaz 65536-ot. Bár ez nem tragikus, mégis kényelmetlen. Mi az oka az eltérésnek?

A szabad tárterület kiszámítását elvégző rutin a \$B37D tárcímen kezdődik (l. „A 64 Belső felépítése” c. könyvet). A számítás előtt a főlöszleges füzérek törlődnek és így ezek a tárcímek felszabadulnak. A számítás menete: a füzérváltozók kezdetéből (\$33/\$34) le kell vonni a tömbök végét (\$31/\$32). A kapott 16 bites egész típusú értéket át kell alakítani lebegőpontos számmá. Az egész típusú számot a rendszer olyan előjeles számként kezeli, melynek értéke – 32768 és 32767 közé esik. Ha ugyanezt a számot pozitív számként értelmezzük, értéke 0 és 65536 közé esik. Amíg a korábbi Commodore gépeken a szabad tárterület nem haladta meg a 32767 byte-ot, addig nem is volt semmi hiba. A Commodore 64-es FRE rutinját úgy kell megváltoztatni, hogy az egész típusú (%) konstans lebegőpontosossá alakításakor a számot pozitív értékűnek tekintse, mint pl. a BASIC sorszámoknál, amelyek köztudottan 32767-nél nagyobb értéket is felvehetnek.

A FRE rutint az alábbiak szerint kell módosítani, és egy olyan területre kell elhelyezni, melyet az interpreter nem használ:

```
0 REM **** P34. ****
1 :
2 :
100 : B38D 4C 55 BF      JMP $BF55
110 : B390 EA           NOP
111 :
112 :
120 : BF55 A5 34         LDA $34
130 : BF57 E5 32         SBC $32
140 : BF59 A2 00         LDX ##00
150 : BF5B 86 0D         STX $0D
160 : BF5D 85 62         STA $62
170 : BF5F 84 63         STY $63
180 : BF61 A2 90         LDX ##90
190 : BF63 4C 49 BC      JMP $BC49
```

Az átmásolást a RAM-ba, majd a rutin betöltését az alábbi kis program végzi el:

```
0 REM **** P35. ****
1 :
2 :
100 FOR I=40960 TO 49151
110 POKE I,PEEK(I):NEXT I
120 A=11*4096+3*256+8*16+13
130 FOR I=A TO A+3
140 READ X:POKE I,X:NEXT I
150 A=11*4096+15*256+5*16+5
160 FOR I=A TO A+16
170 READ X:POKE I,X:NEXT I
180 POKE I,54
200 DATA 76,85,191,234
210 DATA 165,52,229,50,162,0,134,13,133,98,132,99
220 DATA 162,144,76,73,188
```

4.4 Visszatérés a BASIC programba LIST parancs után

A BASIC programban elhelyezett LIST utasítás végrehajtása után a vezérlés visszakérül a rendszerhez, azaz a gép közvetlen üzemmódban folytatja a munkát. Ez meglehetősen zavaró, ha pl. olyan sorokat akarunk listázni, amelyek DEF FN utasítással meghatározott függvényeket tartalmaznak (azért, hogy az éppen feldolgozás alatt álló függvény képlete látható legyen a képernyőn). Hasonlóképpen, azt sem tudjuk megoldani, hogy egy programról egymás után több listát készítsünk a következő ciklussal:

```
FOR I=1 TO 2 : LIST : NEXT
```

Alakítsuk át saját igényeinknek megfelelően a rendszer LIST utasítását! A LIST utasítás végére rakjunk be egy visszaugrást a BASIC melegstartra, azaz helyezzünk el itt egy RETURN utasítást. A LIST végrehajtása előtt tárolnunk kell a programszöveg-mutató aktuális helyzetét – ez ugyanis listázás közben megváltozik –, hogy listázás után a megfelelő helyre vissza tudjunk térni. A szükséges rutin végrehajtja a feladatot és visszatér a BASIC-be.

A következő gépi kódú rutin tartalmazza a BASIC-ROM átmásolását is, így elkerüljük a lassú POKE ciklust. A gépi kódú rutint a kazettapufferbe helyeztük, így a SYS 828 utasítással indíthatjuk. Hívás után a LIST utasítást programmegszakítás veszélye nélkül használhatjuk.

```
0 REM **** P36. ****
1 :
2 :
10 : 033C      * = 828      ;KAZETTAPUFFER
20 : 033C      CHRPTR = $7A      ;PROGRAMMUTATO
30 : 033C      CHRGOT = $79      ;AZ UTOLJARA BETOLTOTT KARAKTER
40 : 033C      LIST = $A69C      ;LIST-RUTIN
50 : 033C      LSTVEC = $A042      ;A LIST RUTIN MUTATOJA
60 : 033C      NEXTST = $ABF8      ;A PRG-MUTATO BEALLITASA
70 : 033C      CRLF = $AAD7      ;KOCSI VISSZA
200 : 033C      A2 20      LDX #32      ;32 LAP ATMASOLASA
210 : 033E      A9 A0      LDA #>$A000 ;VISSZA A BASIC KEZDETRE
220 : 0340      A0 00      LDY #0
230 : 0342      B4 22      STY $22
240 : 0344      B5 23      STA $23
250 : 0346      B1 22      LDA ($22),Y ;MASOLO CIKLUS
260 : 0348      91 22      STA ($22),Y
270 : 034A      C8        INY
280 : 034B      D0 F9      BNE LOOP
290 : 034D      E6 23      INC $23
300 : 034F      CA        DEX
310 : 0350      D0 F4      BNE LOOP
320 : 0352      A9 60      LDA #$60      ;RTS-KOD
330 : 0354      8D 14 A7   STA $A714
340 : 0357      A9 EA      LDA #$EA      ;NOP-KOD
350 : 0359      8D BB A6   STA $A6BB
360 : 035C      8D BC A6   STA $A6BC
370 : 035F      A9 6D      LDA #<NEWLST-1 ;AZ UJ LIST CIME
380 : 0361      8D 42 A0   STA LSTVEC
390 : 0364      A9 03      LDA #>NEWLST-1
400 : 0366      8D 43 A0   STA LSTVEC+1
410 : 0369      A9 36      LDA #$36      ;ATKAPCSOLAS A RAM-RA
420 : 036B      B5 01      STA 1
430 : 036D      60        RTS
440 : 036E      A5 7A      NEWLST LDA CHRPTR
450 : 0370      48        PHA      ;PRG-MUTATO TAROLASA
460 : 0371      A5 7B      LDA CHRPTR+1
470 : 0373      48        PHA
480 : 0374      20 79 00   JSR CHRGOT ;A KOVETKEZO KARAKTER BETOLTESE
```

| | |
|---------------------|---|
| 490 : 0377 20 9C A6 | JSR LIST ;LIST VEGREHAJTASA |
| 500 : 037A 20 D7 AA | JSR CRLF ;KOCSI VISSZA |
| 510 : 037D 68 | PLA |
| 520 : 037E 85 7B | STA CHRPTR+1 |
| 530 : 0380 68 | PLA ;A PRG-MUTATO VISSZATOLTESE |
| 540 : 0381 85 7A | STA CHRPTR |
| 550 : 0383 20 F8 AB | JSR NEXTST ;MUTATO A KOVETKEZO UTASITASRA |
| 560 : 0386 4C 79 00 | JMP CHRGT ;AZ UTOLSO KARAKTER BETOLTESE |

A BASIC betöltő

```

0 REM **** P37. ****
1 :
2 :
100 FOR I=828 TO 904
110 READ X:POKE I,X:S=S+X:NEXT
120 DATA 162, 32,169,160,160, 0,132, 34,133, 35,177, 34
130 DATA 145, 34,200,208,249,230, 35,202,208,244,169, 96
140 DATA 141, 20,167,169,234,141,187,166,141,188,166,169
150 DATA 109,141, 66,160,169, 3,141, 67,160,169, 54,133
160 DATA 1, 96,165,122, 72,165,123, 72, 32,121, 0, 32
170 DATA 156,166, 32,215,170,104,133,123,104,133,122, 32
180 DATA 248,168, 76,121, 0
190 IF S<>9613 THEN PRINT"HIBA A DATA SORBAN..!!":END
200 PRINT"OK.!"

```

Futtassuk le az alábbi BASIC programot, miután végrehajtottuk a SYS 828 utasítást.

```

0 REM **** P38. ****
1 :
2 :
100 PRINT "LIST-TEST"
110 LIST 120
120 GOTO 100

```


4.5 Változó sorszámok a GOTO, a GOSUB és a RESTORE utasításokban

Ebben a fejezetben a BASIC interpreter egyik célszerű módosítását ismertetjük. Programjainkban az elágazások, ill. szubrutinhívások megszervezésénél mindig konstans sorszámokat kell megadnunk. Kényelmesebb lenne, ha a hivatkozott sorszámokat a programon belül kiszámíthatnánk vagy beolvashatnánk, mint az alábbi kis programrészletben:

```
100 INPUT "SORSZÁM" ; Z
110 GOTO Z
```

Ezzel a megoldással helyettesíthetnénk a terjedelmes ON...GOTO... utasítást. Ugyanez vonatkozik a GOSUB utasításra, itt sem ártana a kiszámítható ugrási cím!

Ugyancsak jelentős lépés lenne a RESTORE utasításban megadható változó sorszám. Ha több különböző összefüggő adatblokkal dolgozunk, amelyeket többször végig kell olvasnunk, a READ-DATA mutatót mindig az adatok elejére kell állítanunk, majd sok időt pazarolva bizonyos részeket át kell olvasnunk, míg el nem érjük a kívánt adathalmazt. Sokkal egyszerűbb lenne, ha a RESTORE utasításban megadhatnánk egy sorszámot, és az olvasást azonnal a keresett helyen kezdhethetnénk.

A GOTO utasítás megváltoztatása csak néhány POKE utasítást igényel. A sorszám betöltését végző rutint egy olyan rutinnal kell helyettesítenünk, amely beolvassa a kívánt aritmetikai kifejezést, és ebből előállítja az ugrási címet. A GOSUB utasításnál hasonló a helyzet, mivel a GOSUB végrehajtása GOTO – hívással történik.

```
ABA0 20 C0 02 JSR $02C0
02C0 20 8A AD JSR AD8A
02C3 4C F7 B7 JMP $B7F7
```

A szükséges kiegészítést a \$02C0 (704) tárcímre helyeztük, ez ugyanis mindaddig kihasználatlan, amíg a 11-es sprite-tal nem dolgozunk.

Nézzük a módosító programot:

```
0 REM **** P39/1. ****
1 :
2 :
100 FOR I=40960 TO 49151
110 POKE I,PEEK(I):NEXT
120 A=10*4096+8*256+10*16
130 FOR I=A TO A+2
140 READ X:POKE I,X:NEXT
150 A=704
160 FOR I=A TO A+5
170 READ X:POKE I,X:NEXT
180 DATA 32,192,2
190 DATA 32,138,173,76,247,183
```

Ezzel elintéztük a GOTO és a GOSUB utasításokat. A RESTORE megváltoztatása nem ennyire egyszerű, mivel ebben az esetben az alap BASIC nyelv semmiféle sorszám használatát nem engedi meg.

Meg kell különböztetnünk a sorszámot viselő és sorszám nélküli RESTORE utasítást. A helyzet persze nem annyira nehéz, mint ahogyan látszik!

```

0 REM **** P40. ****
1 :
2 :
100 : 02C6 D0 03 BNE $02CB ;VAN MEG KARAKTER
110 : 02C8 4C 1D AB JMP $AB1D ;UGRAS A REGI RESTORE UTASITASRA
120 : 02CB 20 C0 02 JSR $02C0 ;SORSZAM BETOLTESE
130 : 02CE 20 13 A6 JSR $A613 ;A SORSZAMNAK MEGFELELO CIM KIVALASZTASA
140 : 02D1 38 SEC
150 : 02D2 A5 5F LDA $5F ;A CIM (LO)
160 : 02D4 E9 01 SBC #$01 ;1 LEVONASA
170 : 02D6 A4 60 LDY $60 ;A CIM (HI)
180 : 02DB 4C 24 AB JMP $AB24 ;A TOVABBIKABAN AZONOS A REGI RESTORE-RAL

```

A BASIC betöltő:

```

0 REM **** P39/2. ****
1 :
2 :
300 A=2*256+12*16+6
310 FOR I=A TO A+20
320 READ X:POKE I,X:NEXT
330 DATA 208,3,76,29,168,32,192,2,32,19,166
340 DATA 56,165,95,233,1,164,96,76,36,168

```

A következő két sorban közöljük az interpreterrel az új RESTORE rutin kezdőcímét:

```

0 REM **** P39/3. ****
1 :
2 :
400 POKE 40996, 197 : POKE 40997, 2
410 POKE 1, 54

```

A 410-es sorban átkapcsolunk a RAM-ra. Ettől kezdve a RESTORE utasítást háromféleképpen használhatjuk: sorszám nélkül mint eddig, sorszámmal, végül kifejezéssel, amelyet kiértékelve megkapjuk a kívánt sorszámot. Mindhárom esetben a következő READ utasítás az adott sorban a legelső adatnál kezdi az olvasást. Ha rossz sorszámot adtunk meg (pl. olyat, amely nem létezik), nem kapunk hibajelzést, hanem a mutató automatikusan a következő sorra ugrik.

Nézzünk néhány példát, amelyek használhatóak a módosítások után:

```

100 GOTO 200
...
200 RESTORE 10
...
500 RESTORE
...
800 GOSUB A*2+100
...
900 RESTORE X*10+500

```

4.6 A MID\$ utasítás

Az Olvasó előtt biztosan ismert a MID\$ függvény, amellyel egy fűzer valamely részét leválaszthatjuk. Futtassuk le az alábbi néhány soros programot:

```
0 REM **** P41. ****
1 :
2 :
100 A$="COMMODORE"
110 B$=MID$(A$,4,5)
120 PRINT B$
```

A futtatás eredménye:

MODOR

Az alábbiakban bemutatunk egy olyan MID\$ függvényt, amellyel nemcsak kiemelhetjük a fűzer valamely részét, hanem helyettesíthetjük is azt egy tetszőlegesen megadott karaktersorozattal:

```
0 REM **** P42. ****
1 :
2 :
100 A$="COMMODORE"
110 MID$(A$,4,5)="12345"
120 PRINT A$
```

A 110-es sorban az eredeti fűzer öt karakterét a jobb oldalon megadott öt karakterre cseréljük (a 4. pozíciótól, 5 karakter hosszán!).

Az eredmény:

Az új MID\$ nélkül, alap BASIC-ben ugyanezt a műveletet kicsit körülményesebben lehet elvégezni:

```
0 REM **** P43. ****
1 :
2 :
100 A$="COMMODORE"
110 A$=LEFT$(A$,4)+"123"+MID$(A$,8,3)
120 PRINT A$
```

Különösen hasznos a módosított MID\$ függvény a file-kezelésben, fix hosszúságú mezőkből álló rekordok összeállításához. Előzetesen fel kell töltenünk egy fűzert üres karakterrel, amelybe rendre a megfelelő pozícióktól kezdve az új utasítással beleírhatjuk a kívánt mezőket.

A MID\$ utasítást módosító gépi kódú program:

```
0 REM **** P44. ****
1 :
2 :
90 : 033C .OPT P1
91 : ;A MID$ MINT PSZEUDOVALTOZO
92 : ;
93 : ;A MID$(FUZER,POZICIO,HOSSZ)=FUZER-KIFEJEZES
94 : ;
200 : 033C MIDCODE = $CA
```

```

210 : 033C      EXECUT = $308 ;AZ UTASITAS VEGREHAJTASANAK VEKTORA
220 : 033C      CHRGET = $73
230 : 033C      CHRGET = CHRGET+6
240 : 033C      EXECOLD = $AFEF
250 : 033C      VARNAM = $45
255 : 033C      VARADR = $49
260 : 033C      DESCRPT = $64
270 : 033C      TESTSTR = $AD8F
280 : 033C      GETVAR = B0BB
290 : 033C      SETSTR = $AA52
300 : 033C      CHKAUF = $AEFA ;NYITO ZARQJEL
310 : 033C      CHKZU = $AEF7 ;ZARO ZARQJEL
320 : 033C      CHKCOM = $AEFD ;VESSZO
325 : 033C      TEST = $AEFF
330 : 033C      GETBYT = $B79E
340 : 033C      FRMEVL = $AD9E
350 : 033C      ILLQUAN = $B248
352 : 0003      FRESTR = $B6A3
355 : 0003      *= 3
360 : 0004      LAENGE *= ++1
370 : 0005      POSITION *= ++1
372 : 0007      VARSTR *= ++2
375 : 0007      GLEICH = $B2
378 : 0007      ZEIG2 = $50
379 : ;
390 : 033C      *= 828
400 : 033C A9 47      INIT      LDA #<MIDTEST
410 : 033E A0 03      LDY #>MIDTEST
420 : 0340 0D 08 03    STA EXECUT
430 : 0343 0C 09 03    STY EXECUT+1
440 : 0346 60           RTS
450 : 0347 20 73 00    MIDTEST JSR CHRGET
460 : 034A C9 CA      CMP #MIDCODE ;A MID$ KODJA
470 : 034C F0 06      BEQ MIDSTR
480 : 034E 20 79 00    JRS CHRGET
490 : 0351 4C E7 A7    JMP EXECOLD ;AZ EREDETI UTASITAS VEGREHAJTASA
500 : 0354 20 73 00    MIDSTR JSR CHRGET ;A KOVETKEZO JEL
510 : 0357 20 FA AE    JSR CHKAUF ;NYITO ZARQJEL
520 : 035A 20 8B B0    JSR GETVAR ;A VALTOZO BETOLTESE
530 : 035D 85 64      STA DESCRPT
540 : 035F 84 65      STY DESCRPT+1
550 : 0361 85 49      STA VARADR
560 : 0363 84 4A      STY VARADR+1
570 : 0365 20 A3 B6    JSR FRESTR
580 : 0368 A0 00      LDY #0
590 : 036A B1 64      LDA (DESCRPT),Y
600 : 036C 48         PHA ;HOSSZ
610 : 036D F0 2E      BEQ ILL
620 : 036F 20 52 AA    JSR SETSTR ;A FUZER BETOLTESE A TARBA
630 : 0372 A0 01      LDY #1
640 : 0374 B1 49      LDA (VARADR),Y
650 : 0376 85 05      STA VARSTR ;A VALTOZO CIMENEK TAROLASA
660 : 0378 C8         INY
670 : 0379 B1 49      LDA (VARADR),Y
680 : 037B 85 06      STA VARSTR+1
690 : 037D 20 FD AE    JSR CHKCOM
700 : 0380 20 9E B7    JSR GETBYT ;A POZICIO BETOLTESE
710 : 0383 8A         TXA
720 : 0384 F0 17      BEQ ILL
730 : 0386 CA         DEX
740 : 0387 86 04      STX POSITION
750 : 0389 20 79 00    JSR CHRGET
760 : 038C C9 29      CMP #")" ;A KIFEJEZES VEGE
770 : 038E D0 04      BNE NEXT
780 : 0390 A9 FF      LDA #$FF ;MAXIMALIS HOSSZ
790 : 0392 D0 0C      BNE STORE
800 : 0394 20 FD AE    NEXT JSR CHKCOM

```

```

810 : 0397 20 9E B7 JSR GETBYT ;A HOSSZ BETOLTESE
820 : 039A 8A TXA
830 : 039B D0 03 BNE **+5
840 : 039D 4C 48 B2 ILL JMP ILLQUAN
850 : 03A0 85 03 STORE STA LAENGE
860 : 03A2 68 SEC
870 : 03A3 38 SEC
880 : 03A4 E5 04 SBC POSITION
890 : 03A6 C5 03 CMP LAENGE
900 : 03AB B0 02 BCS OK
910 : 03AA 85 03 STA LAENGE
920 : 03AC 20 F7 AE OK JSR CHKZU ;ZARO ZAROJEL
930 : 03AF A9 B2 LDA #GLEICH
940 : 03B1 20 FF AE JSR TEST
950 : 03B4 20 9E AD JSR FRMEVL ;A KIFEJEZES BETOLTESE
960 : 03B7 20 A3 B6 JSR FRESTR
970 : 03BA A0 02 LDY #2
980 : 03BC B1 64 LDA (DESCRPT),Y
990 : 03BE 85 51 STA ZEIG2+1
1000 : 03C0 88 DEY
1010 : 03C1 B1 64 LDA (DESCRPT),Y
1020 : 03C3 85 50 STA ZEIG2
1030 : 03C5 88 DEY
1040 : 03C6 B1 64 LDA (DESCRPT),Y
1050 : 03C8 F0 03 BEQ ILL ;HA 0 AKKOR HIBA
1060 : 03CA C5 03 CMP LAENGE
1070 : 03CC B0 02 BCS OK1
1080 : 03CE 85 03 STA LAENGE
1090 : 03D0 A5 05 OK! LDA VARSTR
1100 : 03D2 18 CLC
1110 : 03D3 65 04 ADC POSITION
1120 : 03D5 85 05 STA VARSTR
1130 : 03D7 90 02 BCC **+4
1140 : 03D9 E6 06 INC VARSTR+1
1150 : 03DB A4 03 LDY LAENGE
1160 : 03DD 88 LOOP DEY
1170 : 03DE B1 50 LDA (ZEIG2),Y ;EGY KARAKTER ATVITELE
1180 : 03E0 91 05 STA (VAGSTR),Y;A FUZERBE
1190 : 03E2 C0 00 CPY #0
1200 : 03E4 D0 F7 BNE LOOP
1210 : 03E6 4C AE A7 JMP #A7AE ;VISSZA AZ INTERPRETER CIKLUSHOZ

```

A program betöltés után a SYS 828 utasítással indítható.

A BASIC betöltőprogram:

```

0 REM **** P45. ****
1 :
2 :
100 FOR I=828 TO 1000
110 READ X:POKE I,X:S=S+X:NEXT
120 DATA 169, 71,160, 3,141, 8, 3,140, 9, 3, 96, 32
130 DATA 115, 0,201,202,240, 6, 32,121, 0, 76,231,167
140 DATA 32,115, 0, 32,250,174, 32,139,176,133,100,132
150 DATA 101,133, 73,132, 74, 32,163,182,160, 0,177,100
160 DATA 72,240, 46, 32, 82,170,160, 1,177, 73,133, 5
170 DATA 200,177, 73,133, 6, 32,253,174, 32,158,183,138
180 DATA 240, 23,202,134, 4, 32,121, 0,201, 41,208, 4
190 DATA 169,255,208, 12, 32,253,174, 32,158,183,138,208
200 DATA 3- 76, 72,178,133, 3,104, 56,229, 4,197, 3
210 DATA 176, 2,133, 3, 32,247,174,169,178, 32,255,174
220 DATA 32,158,173, 32,163,182,160, 2,177,100,133, 81
230 DATA 136,177,100,133, 80,136,177,100,240,211,197, 3
240 DATA 176, 2,133, 3,165, 5, 24,101, 4,133, 5,144
250 DATA 2,230, 6,164, 3,136,177, 80,145, 5,192, 0
260 DATA 208,247, 76,174,167
270 IF S<>19273 THEN PRINT "HIBA A DATA SORBAN..!":END
280 SYS 828:PRINT "OK..!!"

```

Nézzünk egy példát az új utasítás használatára:

```
0 REM *** P46. ***
1 :
2 :
100 DIM A$(10)
110 FOR I=1 TO 10
120 A$(I)="PROBASZOVEG"
130 NEXT
140 FOR I=1 TO 10
150 MID$(A$(I),I,1)=MID$("1234567890",I,1)
160 NEXT
170 FOR I=1 TO 10
180 PRINT A$(I)
190 NEXT
```

Eredményként tíz füzért kapunk, az elsőben az 1. helyre egy 1-es, a másodikban a 2. helyre egy 2-es került.

```
1ROBASZOVEG
P2OBASZOVEG
PR3BASZOVEG
PRO4ASZOVEG
PROB5SZOVEG
PROBA6ZOVEG
PROBAS7ZOVEG
PROBASZ8VEG
PROBASZ09EB
PROBASZOV0G
```

4.7 Az INSTR és a STRING\$ függvények

A legtöbb BASIC interpreterrel ellátott személyi számítógépen van két olyan fűzérfüggvény, amely a C-64-esen hiányzik. Nevezetesen a STRING\$ függvény – amellyel egy tetszőlegesen hosszú fűzért készíthetünk egy megadott ASCII kódú karakterből –, és az INSTR függvény – amely megvizsgálja, hogy egy fűzérben szerepel-e egy másik vagy sem.

A Commodore 64-es fűzérkezelő műveleteinek és BASIC értelmezőjének ismeretében mindkét függvényt előállíthatjuk.

Utasításszóként használjuk a POS és az STR\$ szimbólumokat, de megkülönböztetésül az eredeti hasonló nevű utasításoktól eléjük felkiáltójelet írunk.

Az INSTR függvény szintaktikája a következő:

`I = ! POS (A$, B$, P)`

ahol az A\$ az a fűzér, amiben keresünk, a B\$ amit keresünk, P pedig az a pozíció, ahonnan kezdjük a keresést. Az I változó értéke 0 ha a B\$ fűzér nem fordult elő az A\$-ban, egyébként pedig az első előfordulás pozíciója. A P változó megadása nem kötelező; ha nincs megadva, a keresés a fűzér elején kezdődik. Változó helyett aritmetikai kifejezést is használhatunk.

Nézzünk néhány alkalmazást:

```
PRINT !POS ("ABCDEFGHJK", "D")
4
IF !POS (A$,"J") THEN PRINT "NEM TALALOM!"
A$ = "HAHAHA"
PRINT !POS (A$,"H")
1
X = !POS (A$,"A",5) : PRINT X
6
```

A STRING\$ függvény használata:

```
A$ = !STR$(L,B)
vagy
A$ = !STR$(L,B$)
```

ahol A\$ az előállítandó fűzér, L a hossza, és B az A\$-t alkotó karakter ASCII kódja. Ha B helyett fűzért használunk, akkor a fűzér első karakterének ASCII kódját veszi a függvény.

Nézzünk néhány példát!

```
PRINT !STR $ (20,65)
AAAAAAAAAAAAAAAAAAAAA
A$ = !STR$(10, "x" ) : PRINT A$
xxxxxxxxxx
```

A gépi kódú programot a szabad RAM területen helyeztük el, 51200-as kezdőcímtől.

```

0 REM ****,P47./****
1 :
2 :
70 : C800 .OPT P1
80 : ;AZ INSTRING-FUGGVENY
90 : ;
95 : ;
100 : C800 == $C800
110 : C800 CHKAUF = $AEFA
120 : C800 CHKZU = $AEF7
130 : C800 CHKCOM = $AEFD
140 : C800 FRMEVL = $AD9E
150 : C800 CHKSTR = $ADB7
160 : C800 FRESTR = $B6A7
170 : C800 YFAC = $B3A2
180 : C800 CHRGET = $73
200 : C800 CHRGET = CHRGET+6
205 : C800 GETBYT = $B79B
210 : C800 INTEGER = $B1AA
215 : C800 DESCRIPT = $64
220 : C800 STRADP = $62
225 : C800 ADR2 = $FB
230 : C800 ADR1 = $FB+2
235 : C800 LEN1 = 3
240 : C800 LEN2 = 4
241 : C800 ANZANL = 5
242 : C800 START = 6
243 : C800 TYPFLAG = 13
244 : C800 STRCODE = $C4
245 : C800 ILLQUAN = $B24B
246 : C800 SYNTAX = $AF0B
247 : C800 POSTCODE = $B9
248 : C800 VECTOR = $30A
249 : C800 TEMP = LEN1
250 : C800 A9 0B LDA #<TESTIN
260 : C802 A0 C8 LDY #>TESTIN
270 : C804 8D 0A 03 STA VECTOR
280 : C807 8C 0B 03 STY VECTOR+1
290 : C80A 60 RTS
300 : C80B A9 00 TESTIN LDA #0
310 : C80D 85 0D STA TYPFLAG
320 : C80F 20 73 00 JSR CHRGET
330 : C812 C9 21 CMP #" "
340 : C814 F0 06 BEQ TEST2
350 : C816 20 79 00 JSR CHRGET
360 : C819 4C 8D AE JMP #AEBD
370 : C81C 20 73 00 TEST2 JSR CHRGET
380 : C81F C9 89 CMP #POSTCODE
390 : C821 F0 0A BEQ INSTR
400 : C823 C9 C4 CMP #STRCODE
410 : C825 D0 03 BNE **+5
420 : C827 4C B1 C8 JMP STRING
430 : C82A 4C 0B AF JMP SYNTAX
440 : C82D 20 73 00 INSTR JSR CHRGET
450 : C830 20 FA AE JSR CHKAUF ;ZARUJEL KI
460 : C833 20 9E AD JSR FRMEVL ;A KIFEJEZES BETOLTESE
470 : C836 20 8F AD JSR CHKSTR ;A FUZER VIZSGALATA
480 : C839 A5 64 LDA DESCRIPT
490 : C83B 48 PHA ;A FUZER CIME A VEREMBOL
500 : C83C A5 65 LDA DESCRIPT+1
510 : C83E 48 PHA
520 : C83F 20 FD AE JSR CHKCOM ;VESSZO
530 : C842 20 9E AD JSR FRMEVL ;A MASODIK FUZER
540 : C845 20 A3 B6 JSR FRESTR
550 : C848 F0 64 BEQ ILL ;HOSSZ=0
560 : C84A 85 04 STA LEN2
570 : C84C 86 FB STX ADR2

```


| | | | | | | | |
|--------|------|----|--------------|-----|-----|----------|-----------------------------------|
| 580 : | C84E | B4 | FC | | STY | ADR2+1 | |
| 590 : | C850 | 68 | | | PLA | | |
| 600 : | C851 | A8 | | | TAY | | |
| 610 : | C852 | 68 | | | PLA | | ;AZ ELSO FUZER CIME |
| 620 : | C853 | 20 | AA B6 | | JSR | FRESTR+7 | |
| 630 : | C856 | F0 | 56 | | BEQ | ILL | |
| 640 : | C858 | 85 | 03 | | STA | LEN1 | |
| 650 : | C85A | 86 | FD | | STX | ADR1 | |
| 660 : | C85C | 84 | FE | | STY | ADR1+1 | |
| 670 : | C85E | A2 | 00 | | LDX | #0 | ;ALAPERTELMEZES A 3. PARAMETERNEL |
| 680 : | C860 | 20 | 79 00 | | JSR | CHRGOT | |
| 690 : | C863 | C9 | 2C | | CMP | "", " | |
| 700 : | C865 | D0 | 07 | | BNE | L1 | |
| 710 : | C867 | 20 | 9B B7 | | JSR | GETBYT | |
| 720 : | C86A | 8A | | | TXA | | ;A KEZDO POZICIO |
| 730 : | C86B | F0 | 41 | | BEQ | ILL | |
| 740 : | C86D | CA | | | DEX | | |
| 750 : | C86E | 86 | 06 | L1 | STX | START | ;A FUZERBELI KEZDOPOZICIO |
| 760 : | C870 | 20 | F7 AE | | JSR | CHKZU | |
| 770 : | C873 | A5 | 03 | | LDA | LEN1 | |
| 780 : | C875 | 38 | | | SEC | | |
| 790 : | C876 | E5 | 04 | | SBC | LEN2 | |
| 800 : | C87B | 90 | 28 | | BCC | END | ;AZ EREDMENY 0 |
| 810 : | C87A | 69 | 00 | | ADC | #0 | |
| 820 : | C87C | 85 | 05 | | STA | ANZAHL | ;ELTOLAS |
| 830 : | C87E | A5 | 06 | | LDA | START | |
| 840 : | C880 | 18 | | | CLC | | ;AZ 1 CIM + KEZDOPOZICIO |
| 850 : | C881 | 65 | FD | | ADC | ADR1 | |
| 860 : | C883 | 85 | FD | | STA | ADR1 | |
| 870 : | C885 | 90 | 02 | | BCC | ++4 | |
| 880 : | C887 | E6 | FE | | INC | ADR1+1 | |
| 890 : | C889 | A0 | 00 | L2 | LDY | #0 | |
| 900 : | C88B | B1 | F8 | L3 | LDA | (ADR2),Y | |
| 910 : | C88D | D1 | FD | | CMP | (ADR1),Y | ;A JELEK OSSZEHASONLITAS |
| 920 : | C88F | D0 | 0B | | BNE | 15 | ;A KOVETKEZO POZICIO |
| 930 : | C891 | C8 | | | INY | | |
| 940 : | C892 | C4 | 04 | | CPY | LEN2 | ;MINDEN JELET MEGVIZSGALUNK |
| 950 : | C894 | 90 | F3 | | BCC | L3 | |
| 960 : | C896 | A4 | 06 | | LDY | START | |
| 970 : | C898 | C8 | | | INY | | |
| 980 : | C899 | 4C | A2 B3 L4 | | JMP | YFAC | ;EREDMENY |
| 990 : | C89C | E6 | 06 L5 | | INC | START | |
| 1000 : | C89E | C6 | 05 | | DEC | ANZAHL | |
| 1010 : | C8A0 | D0 | 04 | | BNE | L6 | ;MEG NINCS KESZEN |
| 1020 : | C8A2 | A0 | 00 | END | IDY | #0 | ;NEM TALALOM, NULLA |
| 1030 : | C8A4 | F0 | F3 | | BEQ | L4 | |
| 1040 : | C8A6 | E6 | FD | L6 | INC | ADR1 | |
| 1050 : | C8A8 | D0 | DF | | BNE | L2 | ;A 2. FUZER CIMENЕК A NOVELESE |
| 1060 : | C8AA | E6 | FE | | INC | ADR1+1 | |
| 1070 : | C8AC | D0 | DB | | BNE | L2 | |
| 1080 : | C8AE | 4C | 48 B2 ILL | | JMP | ILLQUAN | |
| 1090 : | C8B1 | 20 | 73 00 STRING | | JSR | CHRGOT | |
| 1100 : | C8B4 | 20 | FA AE | | JSR | CHKALUF | ;ZAROJEL KI |
| 1110 : | C8B7 | 20 | 9E B7 | | JSR | GETBYT+3 | |
| 1120 : | C8BA | 8A | | | THA | | |
| 1130 : | C8BB | 48 | 20 | | PHA | | ;A HOSSZ TAROLASA |
| 1140 : | C8BC | 20 | FD AE | | JSR | CHKCOM | |
| 1150 : | C8BF | 20 | 9E AD | | JSR | FRMEVL | |
| 1160 : | C8C2 | 24 | 0D | | BIT | TYPFLAG | |
| 1170 : | C8C4 | 30 | 0C | | BMI | STR | ;FUZER |
| 1180 : | C8C6 | 20 | AA B1 | | JSR | INTEGER | |
| 1190 : | C8C9 | A5 | 64 | | LDA | DESCRIPT | ;HB |
| 1200 : | C8CB | D0 | E1 | | BNE | ILL | ; >255 |
| 1210 : | C8CD | A5 | 65 | | LDA | DESCRIPT | ;LB |
| 1220 : | C8CF | 4C | DB C8 | | JMP | STR# | |
| 1230 : | C8D2 | 20 | 82 B7 | | JSR | #B7B2 | ;NUMERIKUS (TIPUSKAPCSOLO) |
| 1240 : | C8D5 | F0 | D7 | | BEQ | ILL | ;A HOSSZ NULLA |

| | | | | | |
|-------------|----------|------|-----|------------|-----------------------------|
| 1250 : C8D7 | A0 00 | | LDY | ILL | |
| 1260 : C8D9 | B1 22 | | LDA | (#22),Y | ;AZ ELSO KARAKTER |
| 1270 : C8DB | 85 03 | STR2 | STA | TEMP | |
| 1280 : C8DD | 68 | | PLA | | ;HOSSZ |
| 1290 : C8DE | 20 7D B4 | | JSR | \$B47D | |
| 1300 : C8E1 | A8 | | TAY | | |
| 1310 : C8E2 | F0 07 | | BEQ | STR3 | |
| 1320 : C8E4 | A5 03 | | LDA | TEMP | |
| 1330 : C8E6 | 88 | LOOP | DEY | | |
| 1340 : C8E7 | 91 62 | | STA | (STRADR),Y | ;A FUZER ELOALLITASA |
| 1350 : C8E9 | D0 FB | | BNE | LOOP | |
| 1360 : C8EB | 20 CA B4 | STR3 | JSR | \$B4CA | ;A FUZER ATVITELE A VEREMBE |
| 1370 : C8EE | 4C F7 AE | | IMP | CHKZU | |

A gépi kódú program BASIC betöltője:

```

0 REM **** P48. ****
1 :
2 :
100 FOR I=51200 TO 51440
110 READ X:POKE I,X:S=S+X:NEXT
120 DATA 169, 11,160,200,141, 10, 3,140, 11, 3, 96,169
130 DATA 0,133, 13, 32,115, 0,201, 33,240, 6, 32,121
140 DATA 0, 76,141,174, 32,115, 0,201,185,240, 10,201
150 DATA 196,208, 3, 76,177,200, 76, 8,175, 32,115, 0
160 DATA 32,250,174, 32,158,173, 32,143,173,165,100, 72
170 DATA 165,101, 72, 32,253,174, 32,158,173, 32,163,182
180 DATA 240,100,133, 4,134,251,132,252,104,168,104, 32
190 DATA 170,182,240, 86,133, 3,134,253,132,254,162, 0
200 DATA 32,121, 0,201, 44,208, 7, 32,155,183,138,240
210 DATA 65,202,134, 6, 32,247,174,165, 3, 56,229, 4
220 DATA 144, 40,105, 0,133, 5,165, 6, 24,101,253,133
230 DATA 253,144, 2,230,254,160, 0,177,251,209,253,208
240 DATA 11,200,196, 4,144,245,164, 6,200, 76,162,179
250 DATA 230, 6,198, 5,208, 4,160, 0,240,243,230,253
260 DATA 208,223,230,254,208,219, 76, 72,178, 32,115, 0
270 DATA 32,250,174, 32,158,183,138, 72, 32,253,174, 32
280 DATA 158,173, 36, 13, 48, 12, 32,170,177,165,100,208
290 DATA 225,165,101, 76,219,200, 32,130,183,240,215,160
300 DATA 0,177, 34,133, 3,104, 32,125,180,168,240, 7
310 DATA 165, 3,136,145, 98,208,251, 32,202,180, 76,247
320 DATA 174
330 IF S<>30119 THEN PRINT "HIBA A DATA SORBAN..!!"
340 SYS 51200:PRINT "OK..!!"

```

8

4.8 A Commodore 64 magyarul beszél*

Zavarja Önt, hogy a Commodore 64 csak angol nyelven beszél!

Tanítsuk meg magyarul!

Az alábbi program a magyar beszédre kényszeríti a gépet, vagy legalábbis a hibaüzenetek magyar nyelvű kiírására.

A hibaüzeneteket (1-től 29-ig sorszámozva) elhelyeztük a programban magyarul (az 1-es hibaüzenet pl. a „too many files”). A táblázat (amely megtalálható „A C 64 belső felépítése” c. könyvben) eredeti angol szövegeit egyszerűen helyettesítettük a megfelelő magyar szöveggel.

Ha van assembler programunk, és saját hibaüzeneteket szeretnénk készíteni, ügyeljünk arra, hogy minden szöveget olyan karakterrel kell befejezni, amelynek megfelelő byte 7. bitje magas. Ezt nagyon egyszerűen elérhetjük, ha az utolsó betűt a SHIFT billentyűvel együtt ütjük le.

```
0 : REM **** P49. ****
1 :
2 :
100 :                                .OPT P1
110 :                                ;
120 :                                ;MAGYAR NYELVU HIBAÜZENETEK A C64-EN
130 :                                ;
140 : C000 ERRVEC = $300 ;A HIBAÜZENETEK VEKTORA
150 : C000 ERRADR = $22 ;MUTATO A HIBAÜZENETRE
160 : C000 OLDERR = $A445
170 :                                ;
180 : C000                                == $C000
190 : C000 A9 0B INIT LDA #<NEWERR
200 : C002 A0 C0 LDY #>NEWERR
210 : C004 8D 00 03 STA ERRVEC
220 : C007 8C 01 03 STY ERRVEC+1
230 : C00A 60 RTS
235 :                                ;
240 : C00B 8A NEWERR TXA ;HIBASZAM
250 : C00C 10 03 BPL ERR1
260 : C00E 4C 43 C0 JMP READY
270 : C011 0A ERR1 ASL A ;2-SZER
280 : C012 AA TAX ;INDEXKENT
290 : C013 BD 48 C0 LDA ERRTAB-2,X
300 : C016 85 22 STA ERRADR
310 : C018 BD 49 C0 LDA ERRTAB-1,X
320 : C01B 85 23 STA ERRADR+1
330 : C01D 20 CC FF JSR $FFCC ;CLRCH
340 : C020 A9 00 LDA #0
350 : C022 85 13 STA #13 ;I/O-KAPCSOLO
360 : C024 20 D7 AA JSR $AAD7 ;UJ SOR
370 : C027 20 45 AB JSR $AB45 ;A "?" KIIRASA
380 : C02A A0 00 LDY #0
390 : C02C B1 22 LOOP LDA (ERRADR),Y
400 : C02E 48 PHA
410 : C02F 29 7F AND #$7F
420 : C031 20 47 AB JSR $AB47
430 : C034 C8 INY
440 : C035 68 PLA
450 : C036 10 F4 BPL LOOP
460 : C038 20 7A A6 JSR $A67A ;FOLYTATAS
470 : C03B A4 3A LDY $3A
480 : C03D C8 INY
```

* Az eredeti szövegben „németül” beszél (a ford. megjegyzése).

```

490 : C03E F0 03      BEQ  READY
500 : C040 20 C2 BD    JSR  $BDC2  ;A SORSZAMBAN
510 : C043 A9 BB      READY LDA  #< KESZ
520 : C045 A0 C2      LDY  #> KESZ
530 : C047 20 78 A4    JSR  $A478
540 : C04A 86 C0 96    ERRTAB .WORD F1,F2,F3,F4,F6,F7,F8
550 : C05A 13 C1 2B    .WORD F9,F10,F11,F12,F13,F14,F15
560 : C068 89 C1 9F    .WORD F16,F17,F18,F19,F20,F21,F22
570 : C076 15 C2 29    .WORD F23,F24,F25,F26,F27,F28,F29,F30
580 :                  ;
590 : C086 54 55 4C F1  .ASC  "TUL SOK FILE !  "
600 : C096 46 49 4C F2  .ASC  "FILE NYITVA"
610 : C0A1 41 20 43 F3  .ASC  "A CS.NINCS NYITVA"
620 : C0B2 41 20 46 F4  .ASC  "A FILE NEM LETEZIK  "
630 : C0C6 41 20 48 F5  .ASC  "A KESZULEK NINCS JELEN"
640 : C0DC 4E 45 4D F6  .ASC  "NEM BEMENETI FILE "
650 : C0EE 4E 45 4D F7  .ASC  "NEM KIMENETI FILE "
660 : C100 41 20 46 F8  .ASC  "A FILENEV HIANYZIK "
670 : C113 49 4C 4C F9  .ASC  "ILLEGALIS KESZULEKSZAM  "
680 : C128 4E 45 58 F10 .ASC  "NEXT FOR NELKUL  "
690 : C13C 46 4F 52 F11 .ASC  "FORMAI HIBA !"
700 : C149 52 45 54 F12 .ASC  "RETURN GOSUB NELKUL  "
710 : C15E 4E 49 4E F13 .ASC  "NINCS TOBB DATA  "
720 : C16E 49 4C 4C F14 .ASC  "ILLEGALIS ERTEK !  "
730 : C180 54 55 4C F15 .ASC  "TUL NAGY!"
740 : C189 42 45 54 F16 .ASC  "BETELT A TARTERULET ! "
750 : C19F 41 20 53 F17 .ASC  "A SOR HIANYZIK"
760 : C1AD 49 4C 4C F18 .ASC  "ILLEGALIS INDEX  "
770 : C1BE 55 4A 52 F19 .ASC  "UJRADIMENZIONALT TOMB  "
780 : C1D8 4F 53 5A F20 .ASC  "OSZTAS NULLAVAL !  "
790 : C1EB 49 47 59 F21 .ASC  "IGY NEM HASZNALHATO !"
800 : C200 48 49 42 F22 .ASC  "HIBAS VALTOZOTIPUS ! "
810 : C215 54 55 4C F23 .ASC  "TUL HOSSZU SZOVEG !  "
820 : C229 48 49 42 F24 .ASC  "HIBAS ADATTIPUS ERKEZETT ! "
830 : C244 41 20 48 F25 .ASC  "A KIFEJEZES TUL OSSZETETT !  "
840 : C261 41 20 50 F26 .ASC  "A PRG.NEM FOLYTATHATO !  "
850 : C27B 4E 45 4D F27 .ASC  "NEM DEFINIALT FUGGVENY !  "
860 : C294 41 5A 4F F28 .ASC  "AZONOSSAGELTERES !  "
870 : C2AA 54 4F 4C F29 .ASC  "TOLTESHIBA"
880 : C2B4 42 52 45 F30 .ASC  "BREAK  "
890 : C2BB 0D          KESZ  .BYT 13  ;CRLF
900 : C2BC 4B 45 53    .ASC  "KESZ .  "
910 : C2C3 0D 00      .BYT 13,0

```

A BASIC betöltőprogram:

```

0 REM **** P50. ****
1 :
2 :
100 FOR I=49152 TO 49860
110 READ X:POKE I,X:I=8+X:NEXT
120 DATA 169, 11,160,192,141, 0, 3,140, 1, 3, 96,138
130 DATA 16, 3, 76, 67,192, 10,170,189, 72,192,133, 34
140 DATA 189, 73,192,133, 35, 32,204,255,169, 0,133, 19
150 DATA 32,215,170, 32, 69,171,160, 0,177, 34, 72, 41
160 DATA 127, 32, 71,171,200,104, 16,244, 32,122,166,164
170 DATA 58,200,240, 3, 32,194,189,169,187,160,194, 32
180 DATA 120,164,134,192,150,192,161,192,178,192,198,192
190 DATA 220,192, 28,192, 0,193, 19,193, 43,193, 60,193
200 DATA 73,193, 94,193,110,193,128,193,137,193,159,193
210 DATA 173,193,190,193,216,193,235,193, 0,194, 21,194
220 DATA 41,194, 68,194, 97,194,123,194,148,194,170,194
230 DATA 180,194, 84, 85, 76, 32, 83, 79, 75, 32, 70, 73
240 DATA 76, 69, 32, 33, 32,160, 70, 73, 76, 69, 32, 78
250 DATA 89, 73, 84, 86,193, 65, 32, 67, 83, 46, 78, 73
260 DATA 78, 67, 83, 32, 78, 89, 73, 84, 86,193, 65, 32

```

```

270 DATA 70, 73, 76, 69, 32, 78, 69, 77, 32, 76, 69, 84
280 DATA 69, 90, 73, 75, 32, 160, 65, 32, 75, 69, 83, 90
290 DATA 85, 76, 69, 75, 32, 78, 73, 78, 67, 83, 32, 74
300 DATA 69, 76, 69, 206, 78, 69, 77, 32, 66, 69, 77, 69
310 DATA 78, 69, 84, 73, 32, 70, 73, 76, 69, 160, 78, 69
320 DATA 77, 32, 75, 73, 77, 69, 78, 69, 84, 73, 32, 70
330 DATA 73, 76, 69, 160, 65, 32, 70, 73, 76, 69, 78, 69
340 DATA 86, 32, 72, 73, 65, 78, 89, 90, 73, 75, 160, 73
350 DATA 76, 76, 69, 71, 65, 76, 73, 83, 32, 75, 69, 83
360 DATA 90, 85, 76, 69, 75, 83, 90, 65, 77, 32, 160, 78
370 DATA 69, 88, 84, 32, 70, 79, 82, 32, 78, 69, 76, 75
380 DATA 85, 76, 32, 160, 70, 79, 82, 77, 65, 73, 32, 72
390 DATA 73, 66, 65, 32, 161, 82, 69, 84, 85, 82, 78, 32
400 DATA 71, 79, 83, 85, 66, 32, 78, 69, 76, 75, 85, 76
410 DATA 32, 160, 78, 73, 78, 67, 83, 32, 84, 79, 66, 66
420 DATA 32, 68, 65, 84, 65, 160, 73, 76, 76, 69, 71, 65
430 DATA 76, 73, 83, 32, 69, 82, 84, 69, 75, 32, 33, 160
440 DATA 84, 85, 76, 32, 78, 65, 71, 89, 161, 66, 69, 84
450 DATA 69, 76, 84, 32, 65, 32, 84, 65, 82, 84, 69, 82
460 DATA 85, 76, 69, 84, 32, 33, 160, 65, 32, 83, 79, 82
470 DATA 32, 72, 73, 65, 78, 89, 90, 73, 203, 73, 76, 76
480 DATA 69, 71, 65, 76, 73, 83, 32, 73, 78, 68, 69, 88
490 DATA 32, 160, 85, 74, 82, 65, 68, 73, 77, 69, 78, 90
500 DATA 73, 79, 78, 65, 76, 84, 32, 84, 79, 77, 66, 32
510 DATA 32, 32, 32, 160, 79, 83, 90, 84, 65, 83, 32, 78
520 DATA 85, 76, 76, 65, 86, 65, 76, 32, 33, 32, 160, 73
530 DATA 71, 89, 32, 78, 69, 77, 32, 72, 65, 83, 90, 78
540 DATA 65, 76, 72, 65, 84, 79, 32, 161, 72, 73, 66, 65
550 DATA 83, 32, 86, 65, 76, 84, 79, 90, 79, 84, 73, 80
560 DATA 85, 83, 32, 33, 160, 84, 85, 76, 32, 72, 79, 83
570 DATA 83, 90, 85, 32, 83, 90, 79, 86, 69, 71, 32, 33
580 DATA 160, 72, 73, 66, 65, 83, 32, 65, 68, 65, 84, 84
590 DATA 73, 80, 85, 83, 32, 69, 82, 75, 69, 90, 69, 84
600 DATA 84, 32, 33, 160, 65, 32, 75, 73, 70, 69, 74, 69
610 DATA 90, 69, 83, 32, 84, 85, 76, 32, 79, 83, 83, 90
620 DATA 69, 84, 69, 84, 84, 32, 33, 32, 160, 65, 32, 80
630 DATA 82, 71, 46, 78, 69, 77, 32, 70, 79, 76, 89, 84
640 DATA 65, 84, 72, 65, 84, 79, 32, 33, 32, 32, 160, 78
650 DATA 69, 77, 32, 68, 69, 70, 73, 78, 73, 65, 76, 84
660 DATA 32, 70, 85, 71, 71, 86, 69, 78, 89, 32, 33, 160
670 DATA 65, 90, 79, 78, 79, 83, 83, 65, 71, 69, 76, 84
680 DATA 69, 82, 69, 83, 32, 33, 32, 32, 32, 160, 84, 79
690 DATA 76, 84, 69, 83, 72, 73, 66, 193, 66, 82, 69, 65
700 DATA 75, 32, 160, 13, 75, 69, 83, 90, 32, 46, 32, 13
710 DATA 0
720 IF S<>59652 THEN PRINT "HIBA A DATA SORBAN..!":END
730 SYS 49152:PRINT "OK.!"
KESZ .

```

4.9 A nyugat-berlini Ku'damm óra a CBM 64-esen

A Kurfürstendammon kószáló tájékozatlan turisták többsége elképedve áll meg egy, a zöld sétány közepére helyezett állvány előtt, melyen sárga és piros lámpák villognak. A látvány első pillantásra fénykavalkádként hat, azután hirtelen összeáll a kép egy furcsa órává, amely sajátos módon mutatja a pontos időt.

A következő program egy szakasztott ilyen furá órát varázsol a Commodore 64-es képernyő-jére. (Az óra vázlatos rajza a 69. oldalon látható.)

Természetesen a különleges óra csak színes TV-n mutat jól. A programban alkalmazott ötletek további lépést jelentenek a Commodore 64-es „képességeinek” felderítésében. Segédeszközként a beépített valós idejű órát és a sprite-okat használtuk fel, ill. természetesen a karaktergrafikát.

A részletes dokumentációval lehetőséget kínálunk arra, hogy az Olvasó a program egyes részeit a saját céljaira felhasználhassa.

A program változói:

| | |
|---------|---|
| A\$ | A pontos idő (input) |
| AP | A karaktergrafika tömbmutatója |
| C | A valós idejű óra báziscíme |
| V | A videovezérlő báziscíme |
| S1 | A 0. sprite pontmintájának címe |
| S3 | Az 1. sprite pontmintájának címe |
| F1 | = 0 a másodperc első fele = 1 a másodperc második fele |
| F2 | = 1 az óra indítása |
| FL | = 0 délelőtt = 1 délután |
| H | Óraszámláló |
| M | Percszámláló |
| S | Másodpercszámláló |
| T | Tizedmásodperc-számláló |
| H\$ | A H változó füzéralakja |
| M\$ | Az M változó füzéralakja |
| S\$ | Az S változó füzéralakja |
| ME\$(X) | Az egyes percek szimbólumgrafikája |
| MZ\$(X) | A tízpercek szimbólumgrafikája |
| SE\$(X) | Az egyes órák szimbólumgrafikája |
| SZ\$(X) | A „tízórák” szimbólumgrafikája |

A programlépések magyarázata:

| | |
|------|---|
| 5–10 | A változók kezdőértékének beállítása |
| 15 | A keret színe fekete, a sprite-okat kikapcsoljuk – erre a program leállítása után és újraindítása előtt van szükség, egyébként a pontos idő beolvasása közben mozognának. |
| 30 | A regiszter 7. bitjét magasra kell állítani, hogy az óra állása 0 legyen. |
| 35 | A valós idejű óra ütemét 50 Hz-re állítjuk (a 7. bit értéke a regiszterben 1). |

- Az óra ütemét a hálózati feszültség szabályozza, ez Európában általában 50 Hz, Amerikában 60 Hz (itt a megfelelő bitet alacsonyra kell állítani).
- 40-50 A pontos idő beolvasása billentyűzetről.
- 60-80 A beolvasott füzért numerikus részekre kell szétszedni.
- 90-100 Mivel az óra 12 órás ciklusban működik az AM/PM jelzőt (az óraregiszter 7. bitje) az adatbevitelnek megfelelően be kell állítani. (Az AM/PM a latin „ante/post meridiem” kifejezés rövidítése, jelentése délelőtt/délután. Az időszakot számos országban így jelölik.)
- 110-160 A kapott értékek egyes és tízes részét szétválasztjuk és a megfelelő regiszterekbe töltjük. Az óraregiszter BCD (Binary Coded Decimal) kódban dolgozik, ebben a kódrendszerben minden félbyte által tartalmazott legnagyobb számérték a 9.
- 161-164 Sprite-adatok a kör kitöltéséhez.
- 165-167 A kör szegélyének sprite-adatai.
- 168-169 A sprite-adatok betöltése.
- 170-171 A sprite-mutatókat beállítjuk, a sprite-okat bekapcsoljuk, kétszeresre nagyítjuk, meghatározzuk a képernyőpozíciójukat és színüket.
- 179-196 Előállítjuk a négyszögletes mező keretét.
- 198-224 A percek és órák kijelzésére szolgáló tömbök értékének és színének beállítása. Nagyobb tárigényű, de gyorsabb megoldás, mint a folyamatos rajzolás.
- 225-229 Félmásodperces időközönként a sprite és a kitöltött kör felváltva kapcsol ki és be.
- 230-290 A valós idejű óra értékét leolvassuk és átalakítjuk decimális számmá. A beolvasott óra értékéhez (az AM/PM kapcsoló értékétől függően – FL) 12-t hozzáadunk, hogy 24 órás kijelzést kapjunk.
- 310-350 A képernyő jobb alsó sarkába kiírjuk a megfelelően átalakított értéket.
- 380-410 A tömbben tárolt perc, ill. óra rajzolatának előállítása.
Mivel minden kombinációhoz egyetlen pontminta tartozik, az óráról beolvasott érték a tömb indexét adja.
A kijelzésre csak percenként, ill. az óra átállása után van szükség (370-, 360-as sorok).

Elnézést kérünk a programlista minőségéért, de a speciális karakterek miatt nem használhatunk jobb nyomtatót.

```
0 REM **** P51. ****
1 :
2 :
3 PRINT "MÉDUM H20(11)
10 C=36280+V-33248:S1=704:83=632:IF2=1
15 POKE V+83,0:POKE V+81,0:PRINT "MÉDUM
30 POKE C+7,PEEK(C+7) AND 127
35 POKE C+8,PEEK(C+8) OR 128
40 PRINT "KEREM A PONTOS IDOT ?"
50 INPUT "FORMATUM HHMM":AS:H=VAL(LEFT$(AS,2))
60 H=VAL(LEFT$(AS,2))
70 S=0
85 H=VAL(RIGHT$(AS,2))
90 IF H>23 THEN 40
95 IF H=12 THEN 110
100 IF H>11 THEN H=H+60
110 POKE C+3,16+INT(H/10)+H-INT(H/10)*10
120 IF M>59 THEN 40
130 POKE C+2,16+INT(M/10)+M-INT(M/10)*10
140 IF S>59 THEN 40
150 POKE C+1,16+INT(S/10)+S-INT(S/10)*10
160 POKE C,0
```

```

161 DATA 0,0,0,0,128,0,3,255,192,7,255,224,15,255,240,31,255,248,31,255,248
162 DATA 31,255,248,31,255,248,63,255,252,63,255,252,63,255,252,31,255,248
163 DATA 31,255,248,15,255,248,15,255,248,7,255,240,3,255,224,3,255,192
164 DATA 0,255,0,0,0,0
165 DATA 1,255,128,7,128,224,12,0,48,24,0,24,48,0,12,48,0,6,86,0,6,86,0,6
166 DATA 96,0,6,192,0,3,192,0,3,192,0,3,96,0,6,96,0,6,48,0,6,48,0,6
167 DATA 24,0,12,12,0,24,12,0,48,7,0,224,3,255,128
168 FOR I=61 TO 81+62:READ A:POKE I,A:NEXT
169 FOR I=53 TO 53+62:READ A:POKE I,A:NEXT
170 POKE 2048,1:POKE 2041,13:POKE V+32,2:POKE V+33,8:POKE V+21,3:POKE V+23,9
171 POKE V+39,7:POKE V+40,0:POKE V,160:POKE V+2,160:POKE V+1,59:POKE V+3,59:POKE
V+28,3
178 PRINT "0123456789"
180 PRINT "ABCDEFGHIJ"
181 PRINT "KLMNOPQR"
182 PRINT "STUVWXYZ"
183 PRINT "01234567"
184 PRINT "89ABCDEFGHIJ"
185 PRINT "KLMNOPQRST"
186 PRINT "UVWXYZ0123"
187 PRINT "456789ABCD"
188 PRINT "EFGHIJKL"
189 PRINT "MNOPQRSTU"
190 PRINT "VWXYZ01234"
191 PRINT "56789ABCDEFGHIJ"
192 PRINT "KLMNOPQRSTU"
193 PRINT "VWXYZ01234"
194 PRINT "56789ABCDEFGHIJ"
195 PRINT "KLMNOPQRSTU"
196 PRINT "VWXYZ01234"
197 PRINT "56789ABCDEFGHIJ"
198 S2*(0)="0123456789"
199 S2*(1)="ABCDEFGHIJ"
200 S2*(2)="KLMNOPQR"
201 S2*(3)="STUVWXYZ"
202 S2*(4)="01234567"
203 SE*(0)="89ABCDEFGHIJ"
204 SE*(1)="KLMNOPQRST"
205 SE*(2)="UVWXYZ0123"
206 SE*(3)="456789ABCD"
207 SE*(4)="EFGHIJKL"
208 M2*(0)="MNOPQRSTU"
209 M2*(1)="VWXYZ01234"
210 M2*(2)="56789ABCDEFGHIJ"
211 M2*(3)="KLMNOPQRSTU"
212 M2*(4)="VWXYZ01234"
213 M2*(5)="56789ABCDEFGHIJ"
214 M2*(6)="KLMNOPQRSTU"
215 M2*(7)="VWXYZ01234"
216 M2*(8)="56789ABCDEFGHIJ"
217 M2*(9)="KLMNOPQRSTU"
218 M2*(10)="VWXYZ01234"
219 M2*(11)="56789ABCDEFGHIJ"
220 ME*(0)="EFGHIJKL"
221 ME*(1)="MNOPQRSTU"
222 ME*(2)="VWXYZ01234"
223 ME*(3)="56789ABCDEFGHIJ"
224 ME*(4)="KLMNOPQRSTU"
225 ON F1 GOTO 228
226 IF PEEK(C)>4 THEN POKE V+39,7:GOTO 225
227 POKE V+39,11:F1=1:GOSUB 230:GOTO 310
228 IF PEEK(C)<5 THEN 228
229 F1=0:GOTO 225
230 H=PEEK(C+3):M=PEEK(C+2):S=PEEK(C+1):T=PEEK(C)
240 FL=1
250 IF H>32 THEN H=H-128:FL=0
260 H=INT(H/16)*10+H-INT(H/16)*10:ON FL GOTO 260
265 IF H=12 THEN 285

```


1. FEJEZET

ELŐSZÓ

A Commodore 64 naponta több ezernyi barátot szerez magának az egész világon. Ez a kis számítógépes rendszer nemcsak korlátlan lehetőséget biztosít a hobby-programozók számára, hanem minden további nélkül alkalmazható kisebb kereskedelmi, adminisztrációs és tudományos műszaki feladatok megoldásában is. Éppen ebben mutatkozik meg a „Tippek és Trükkök” c. könyv hasznossága. A Klaus Gerits, Lothar Englisch és Michael Angerhausen szerzői együttesünk ismét mélyrehatólt a lehetséges trükkök tanulmányozásában, miközben számtalan tippet adnak a C 64-es mind jobb kihasználásához. Az említett szerzők célkitűzése az volt, hogy példák és mintaprogramok alapján ismertessék a Commodore 64 alkalmazási lehetőségeit. A könyv nagy segítséget nyújt abban, hogy a Commodore 64 adottságait messzemenően kihasználhassuk.

Mindhárom szerző úgy ismeri a Commodore 64-est, mint a tenyerét. Klaus Gerits nevéhez fűződik a 64 IEC-BUS és a MAXI 64 kifejlesztése, Lothar Englisch pedig a PROFI-MON 64 és a PROFI-ASS 64 közismert programok alkotója. Mindketten behatóan ismerik a C 64-es műszaki és rendszertulajdonságait. Eredményeik közé tartozik a USER-port tervezetének kidolgozása és a CP/M-modul beható leírása, a minden részletre kiterjedően dokumentált BIOS-listával együtt. Michael Angerhausen – az adatbank-szakértőnk – a file-kezelés kapcsán foglalkozott behatóan a Commodore 64-essel. Többek között azt is bemutatja, hogy milyen eljárással lehet a BASIC 2.0. változat alatt a relatív file-okat felépíteni.

Néhány hónap alatt elfogyott a „Tippek és Trükkök” c. könyv első kiadása. Az új kiadást megelőzően a szerzők természetesen megragadták az alkalmat a közkedvelté és keresetté vált könyv átdolgozására és egyes részeinek kibővítésére. Olvasóink ezt a második, átdolgozott kiadást tarthatják a kezükben.

A „Tippek és Trükkök”-ben foglalt összes gépi programhoz és rutinhoz egy-egy BASIC betöltő programot (BASIC-Loadert) is mellékelünk. Ezeket az egyszerű BASIC programokat azok is könnyen betölthetik, akik gépi kódban nem tudnak programozni és így mindenki gond nélkül használhatja a gépi nyelven írt programokat.

A mindenkor BASIC betöltő ellenőrzi a gépi kódú programot, és kiszűri az adatbevitelből eredő hibákat.

Jó szórakozást kívánunk a „Tippek és Trükkök” c. könyvben foglalt feladatokhoz!

Düsseldorf, 1983. augusztus

A Szerzők

5. FEJEZET

A COMMODORE NEM CSAK A BASIC-ET ISMERI

5.1 Programozás FORTH nyelven

Mit jelent a FORTH? Ezt a kérdést általában azok teszik fel, akik eddig csak BASIC vagy assembler nyelven programoztak. Önök közül is biztosan sokan megkérdezik; „Minek tanuljak meg egy új programnyelvet, amikor a Commodore-om kiválóan érti a BASIC-et?” A kérdés jogosnak látszik mindaddig, míg meg nem ismertük az új nyelv szolgáltatotta lehetőségeket. Akkor különösen nem szabad elzárkózni az új nyelvek megtanulása elől, ha a számítógépünk többféle nyelven programozható.

Aki huzamosabb ideje készít BASIC nyelvű programokat, észreveszi, hogy néhány feladat egyáltalán nem, vagy csak igen nehézkesen oldható meg ezen a nyelven. Nagyobb gépeken általában többféle nyelven programozhatunk. A FORTRAN-t főleg matematikai eljárások, a COBOL-t kereskedelmi célfeladatok, az assemblert főként hardverfüggő, a BASIC-et pedig általános célú feladatok programozási nyelveként használjuk. Van néhány olyan programnyelv, mint a PASCAL vagy pl. az ELAN, amelyeket strukturált programok készítésére fejlesztettek ki. Minden nyelvnek megvannak a saját erős és gyenge pontjai.

A FORTH a programnyelvek legfiatalabb generációjához tartozik, nevezetesen a negyedik (angolul: fourth) generációhoz. Fejlesztői megkísérelték a régi nyelvek előnyeit megtartva, és hátrányaikat lenyelve egy tökéletesebb nyelv kidolgozását.

A FORTH a következőkben felsorolt tulajdonságai alapján alkalmas arra, hogy a személyi számítógépek speciális nyelvéné váljék:

- 1.A FORTH program futtatása kis tárbeli címtartományt igényel. Ezen a nyelven viszonylag nagy, hatékony programokat készíthetünk viszonylag kis tárhatalmú gépeken.
2. Ideálisan alkalmazkodik a gép hardverfelépítéséhez, ami azt jelenti, hogy a programozónak nem kell ismernie a gép hardverkészleteit ahhoz, hogy hatékony programot készíthessen. (Pl. rendkívül kényelmes input/output utasításokkal rendelkezik. Ez a tulajdonsága indokolja gyakori alkalmazását a robot- és vezérléstechnikában.)
3. A fentiekén túl a FORTH rendkívüli előnye, hogy futtatása nem igényel lemezegységet. (Persze ha van, az nem hátrány!)

A FORTH öt fő részből áll:

1. A SZÓTÁR: A FORTH programnyelv néhány, a gépi nyelvhez közelálló FORTH utasításból áll, amelyekből a programozó további utasításokat definiálhat. Mindenki a saját igényeihez igazított programozási nyelvvé dolgozhat, ezzel leegyszerűsítve bizonyos feladattípusok megoldását. A saját célra kialakított utasítások szótárát, a végrehajtási útmutatóval együtt hozzákapszolhatjuk a FORTH állandó utasításkészletéhez.
2. A VEREM: A verem a FORTH legfontosabb építőköve. A verem (stack) fogalmát mindenki ismeri, aki valaha egy kicsit is programozott gépi nyelven, vagy dolgozott HP programozha-

tő zsebszámológéppel. A verem az ún. „LIFO” elven alapszik (ami utoljára bekerült, az kerül ki legelőször – angolul: last in-first out).

Minden művelet, aritmetikai, logikai, összehasonlító művelet az ún. veremelvre épül.

3. Az INTERPRETER: A FORTH is a Commodore 64-esen jól ismert interpreterrel dolgozik. Az editorral (szerkesztő programmal) megírt FORTH programot a megfelelő utasítással közvetlenül elindíthatjuk. A programhibák kiszűrése futás közben történik. Indítás előtt nem kell a programot lefordítani, ami rendkívül időigényes lenne, hanem lehet azonnal futtatni. Ennek persze az a következménye, hogy a programfutást a közbeni értelmezés lassítja. A FORTH esetében ez a lassító tényező nem túl nagy, mivel a FORTH interpreter két részből épül fel: egy ún. szövegértelmezőből és egy címértelmezőből. A szövegértelmező kikeresi a szót a szótárból, és ha megtalálta, elindítja a címértelmezőt, ami feldolgozza az utasításhoz rendelt abszolút címeket. A programok futási ideje ezzel a módszerrel csökkenthető a hagyományos módon interpretált programok futási idejéhez képest.
4. Az ASSEMBLER: Minden FORTH tartalmaz egy assemblert. Az assemblerrel definiálhatjuk a belső gépi kódú rutinokkal végrehajtott, de saját igényünk szerint kialakított utasításokat. Az ilyen programozástechnikai módszer szoros I/O kapcsolatot követel a külvilággal. A FORTH bizonyos értelemben hasonló az assemblerhez, éppen olyan gyors, de könnyebb elsajátítani.
5. A TÁR: A tár minden programnyelv számára fontos, ez alól a FORTH sem kivétel. A FORTH táркеzelésének speciális tulajdonsága, hogy valahányszor megváltoztatjuk a tár egy blokkjának tartalmát, a FORTH egyidejűleg módosítja a lemez megfelelő blokkjának tartalmát is, ha a lemezegység be van kapcsolva.

5.2 A FORTH és a BASIC összehasonlítása

Ebben a fejezetben bemutatunk egy FORTH és egy BASIC programot, amelyek azonos feladatot oldanak meg. Így egyrészt összehasonlíthatjuk a két nyelvet, másrészt érzékelhetjük a FORTH nyelv előnyeit a BASIC-hez képest.

A FORTH program megírásához szükségünk van a nyelv minimális ismeretére.

Az 5.1 fejezetben említettük, hogy a verem nagyon fontos szerepet játszik a FORTH nyelvben, és a veremmelvet a HP zsebszámológép működési elvéhez hasonlítottuk.

Számítsuk ki a $(2 + 3) * (4 + 5)$ kifejezést HP zsebszámológépen. A gombok megnyomásának sorrendje:

2 <ENTER> 3 + 4 <ENTER> 5 + *

A kép kissé zavarosnak tűnik első látásra, hiszen zárójeleket nem használhatunk.

Amikor megnyomtuk a 2-t, majd az ENTER-t, a 2 szám bekerült a verem tetejére. Megnyomva a 3-at, a 2 egy hellyel lejjebb csúszott, és fölé került a 3.

```
A VEREM: TETEJE 3
              2
              -
              -
              -
              -
              -
              ALJA -
```

A + megnyomásakor a gép elvégzi az összeadást, és az összeget a verem tetejére helyezi.

A + művelethez leemeli a verem tetején talált két számot, ezeket összeadja, és az eredményt visszat teszi a verembe:

```
A VEREM: TETEJE 5
              -
              -
              -
              -
              -
              ALJA -
```

Ha megnyomjuk a 4-et, az 5 egygel lejjebb csúszik, és a 4 kerül a verem tetejére.

```
A VEREM: TETEJE 4
              5
              -
              -
              -
              -
              ALJA -
```

Ha ismét megnyomjuk az 5-öt, a folyamat megismétlődik:

```

A VEREM:  TETEJE 5
              4
              5
              -
              -
              -
              ALJA -

```

A + művelethez ismét kiemeli a két legfelső számot a veremből, majd az eredményt visszateszi a verembe:

```

A VEREM:  TETEJE 9
              5
              -
              -
              -
              -
              ALJA -

```

Az utolsó művelet a szorzás. Elvégzése hasonló az eddigiekhez:

```

A VEREM:  TETEJE 45
              -
              -
              -
              -
              -
              ALJA -

```

Kissé hosszadalmas lenne, ha nekünk is így kellene számolnunk, de a zsebszámológépek, sőt mindezidáig a számítógépek is így működnek. Azok, akik valaha assembler nyelven programoztak, rendkívül könnyen elsajátítják a FORTH nyelvet. A többieknek sem okozhat különösebb gondot, hiszen hihetetlenül egyszerű nyelv, a tanulásban az egyetlen nehézséget az okozhatja, hogy ez a nyelv egyetlen másikhoz sem hasonlítható.

Nézzünk meg közelebbről egy néhány soros FORTH programot, amelyből egyrészt kiderül a verem sajátos szerepe, másrészt az, hogy hogyan kell a FORTH szavakat definiálni.

A program feladata egy szám köbének meghatározása. Miután erre a célra nincs külön utasítás, definiálnunk kell a köbreemelést:

```

:KOB      (EZT A SZÓT AKARJUK DEFINIÁLNI)
DUP DUP  (A SZÁMOT KÉTSZER BEIRJUK A VEREMBE)
* *      (A VEREM TARTALMAT KÉTSZER MEGSZOROZZUK ÖNMAGÁVAL)
;

```

Elemezzük a programot! A kettőspont közli a FORTH-szal, hogy egy szót definiálunk. Ha nem írunk kettőspontot, a FORTH a következő kérdést tenné fel:

KOB?

ezzel jelezve, hogy ez a szó nem szerepel a szótárában. A kettősponttól kezdve a pontosvesszőig terjedő programrészt a FORTH a szó definíciójának tekinti, értelmezi, majd a következő kedves üzenetet küldi:

OK

ezzel jelezve, hogy az új szót elhelyezte a szótárában, azaz a KOB utasítást beépíthetjük a programunkba. Ha nem tároljuk lemezen a bővített szótárat, természetesen az új szó csak a gép kikapcsolásáig él, de ha igen, bármikor dolgozhatunk a kibővített szótárral.

A DUP utasítás egy másolatot készít a verem tetején levő értékről, és az előző értéket egy hellyel lejjebb csúsztatva, ezt is berakja a verembe. Mivel egy szám köbét akarjuk kiszámítani, a számról két másolatot kell készítenünk. Ha a verem tetején 5 volt, a verem tartalma a következőképpen módosul:

```
A VEREM:  TETEJE 5
              5
              5
              -
              -
              -
              ALJA -
```

A köbreemeléshez összesen két szorzást kell elvégeznünk, amit a FORTH számára két csillaggal jelölünk. A szorzások végrehajtása után az eredmény a verem tetejére kerül, és készen is vagyunk.

Jegyezzük meg, hogy a definiálás során műveletvégzés nincs. A pontosvessző a FORTH-ot ún. fordító üzemmódra kapcsolja. Fordító üzemmódban sorra elemzi a definícióban előforduló utasításszavakat, és megjegyzi, hogy mi az egyes szavak címe a szótárban. Ha program-sorban használjuk az új utasítást, a FORTH meghívja a definícióban szereplő utasításokat, azaz végrehajtja az új szóban kijelölt műveleteket.

A pont az alábbi utasításszavak mögött azt jelenti, hogy a művelet eredményét a verem tetejére kell tenni.

Próbáljuk ki az általunk definiált új műveletet:

Bevitel *a FORTH üzenete:*

```
5 KOB.           125 OK
1 KOB.           1 OK
- 15 KOB.       - 3375 OK
```

Amint láttuk, rendkívül egyszerű az új szavak definiálása, és ez a módszer hatékonyvá teszi a programozási munkát, hiszen olyan utasításkészlettel dolgozhatunk, amely tökéletesen illeszkedik egy-egy speciális feladat arculatához.

Most hasonlítsuk össze a BASIC-et a FORTH-szal. Az összehasonlítás alapja legyen az a program, amely kiszámítja 0-tól 10-ig az egész számok köbét.

A FORTH programban természetesen felhasználjuk az előbb definiált KOB utasítást.

1. FORTH

:KOB SZAMOK

10 0 (A SZAMOK 0-TOL 10-IG A VEREMBE! LIFO)

DO (A CIKLUS INDUL)

CR I I KOB (AZ I SZAM ES KOBENEK KIIRASA)

LOOP (A CIKLUS VEGE)

;

KOB SZAMOK

0 0

1 1

2 8

3 27

4 64

5 125

6 216

7 343

8 512

9 729 OK

2. BASIC

0 REM **** P52/1. ****

1 :

2 :

10 REM EGY SZAM KOBENEK KISZAMITASA

20 MIN=0:MAX=9

30 FOR I=MIN TO MAX:PRINT I,I*I*I

40 NEXT I

50 END

RUN

0 0

1 1

2 8

3 27

4 64

5 125

6 216

7 343

8 512

9 729

Mindkét program megközelítőleg azonos számú sorból áll, de a FORTH program jóval kevesebb helyet foglal el a tárban mint a BASIC program, és ez még nem minden. Hasonlítsuk össze a futási sebességet:

1. FORTH

```
: IDOTESZT
30000 0      (0-TOL 30000-IG)
DO          (A CIKLUS KEZDETE)
          (URES CIKLUS)
LOOP       (A CIKLUS VEGE)
;          (A DEFINICIO VEGE)
```

2. BASIC

```
0 REM **** P52. ****
1 :
2 :
10 REM IDOTESZT
20 MIN=0:MAX=30000
30 FOR I=MIN TO MAX
40 NEXT I
50 END
KESZ .
```

Az eredmény meglepő:

| Nyelv | Idő |
|-------|-----|
|-------|-----|

| | |
|-------|----------|
| BASIC | kb. 40 s |
|-------|----------|

| | |
|-------|---------|
| FORTH | kb. 4 s |
|-------|---------|

Mindkét program azonos gépen, a Commodore 64-esen futott!

Reméljük ez a kis bevezetés meghozta az Olvasó kedvét a FORTH nyelv tanulmányozásához.

Könnyű vele programozni, rendkívül gyors és kis tárigényű!

Néhány információ azoknak, akik közelebből szeretnének megismerkedni ezzel a programnyelvel:

A hivatkozási (standard) nyelv a FORTH 79. A 64-es változat alatt megírt programokat kisebb változtatásokkal átalakíthatjuk a 79-es változatnak megfelelő programokká.

A FORTH 64 elérhető és azonnal indítható modulok formájában van a lemezen tárolva. Három, egyenként 1024 byte-os lemezpuffert használ (a \$C00-ás címtől kezdve), és saját képernyőtárral dolgozik (a \$0400-as címtől kezdve).

FORTH – A haladó programozók szoftverfejlesztő eszköze.

Érdemes megjegyezni, hogy egyre több professzionális szoftverfejlesztő tér át a FORTH nyelv használatára, pontosan a fent leírt előnyei miatt. A FORTH programfejlesztésre kiválóan alkalmas, hiszen strukturált nyelv, és ez a strukturáltság gyorsítja a fejlesztőmunkát. Ugyanakkor futási sebességben megközelíti a gépi kódú programot. Ma már sok olyan általános szoftver létezik a Commodore 64-esen is, amelyet ezen a nyelven fejlesztettek ki, ilyen például a széles körben elterjedt Calc Result.

A szoftverházak számára nagyon fontos, hogy a FORTH az ún. „szállítható” nyelvek családjába tartozik. Ez azt jelenti, hogy az egyik géptípuson megírt programot nagyon könnyen áttehetjük egy másik géptípusra. Így erősen csökkenthető az egy-egy programcsomag adaptálásához szükséges idő.

5.3. További programnyelvek: PASCAL, LOGO, TURTLE GRAPHICS

A számítógép használhatóságát az általa ismert programnyelvek határozzák meg. A Commodore 64-es nemcsak egyetlen nyelven programozható. A Commodore 64-re kapható CP/M operációs rendszer segítségével csaknem minden nyelvet használhatunk, ami a mikrogépeken egyáltalán elérhető.

Mire szolgálnak az egyes nyelvek?

PASCAL

Ezt a nyelvet eredetileg oktatási célra szánták. Fejlesztője, a svájci N. WIRTH professzor a strukturált programozáshoz kívánt eszközt teremteni. A PASCAL-t ma már sok intézetben és vállalatnál használják alapnyelvként, és számos felsőoktatási intézményben oktatják.

LOGO

Ez a programnyelv többek számára ismeretlen. Elsajátításához még egy gyereknek is csak néhány órára van szüksége, éppen ezért főként a gyerekek programozási nyelve.

TURTLE GRAPHICS (teknősbéka grafika)

A TURTLE ma sem teljesen ismeretlen programnyelv, amely szerkezetileg a PILOT és a LOGO közé sorolható. Ahogyan az a nevében is benne van, elsősorban a grafika kezelését megkönnyítő utasításokat tartalmaz, és mivel a Commodore 64-es gazdag grafikai lehetőségeket tartalmaz, ezen a gépen különösen jól használható programnyelv.

Nézzünk egy kis példát a TURTLE alkalmazására:

```
0 REM **** P53. ****
1 :
2 :
4 REM A KEPERNYO SZINEK BEALLITASA
5 SCREEN COLOR CYAN:A HATTER SZINE CIAN
6 BORDER COLOR CYAN:A KERET SZINE CIAN
7 TURTLE COLOR BLACK:A TEKNOSBEKA SZINE FEKETE
8 REM ATKAPCSOLAS NAGYFELBONTASU GRAFIKARA (HIRES)
9 HIRES
10 REM ATKAPCSOLAS IRASRA
11 PEN UP
12 REM POZICIO,A KEPERNYO BAL FELSO SARKARA
13 MOVE TO 0-0
14 REM AZ IRAS KIKAPCSOLASA
15 PEN DOWN
16 REM POZICIO,A KEPERNYO JOBB ALSO SARKARA
17 MOVE TO 199-319
18 REM VARAKOZOCIKLUS
19 WAIT 300
KESZ .
```

A program egy egyenest rajzol a képernyőre a (0,0) koordinátájú pontból a (199,319) koordinátájú pontba. A rajzolás úgy történik, mintha plotteren dolgoznánk. Amíg a PEN DOWN utasítás érvényben van, a mozgás közben érintett pontok megjelennek a képernyőn (pl. a MOVE utasítás hatására).

Az elmondottakból kitűnik, hogy a Commodore 64-esen igen sokféle programnyelvet használhatunk. Ezek a programnyelvek képessé teszik a gépet arra, hogy minden vonatkozásban felvegye a versenyt a nagyobb rendszerekkel ellátott számítógépekkel.

6. FEJEZET

A CP/M OPERÁCIÓS RENDSZER A COMMODORE 64-ESEN

6.1 Bevezetés a CP/M-be

A CP/M a mikrogépeken leggyakrabban alkalmazott operációs rendszer, amely a maga nemében egyedülálló. Nem egyszer előfordult már a számítógépek történetében, hogy egy operációs rendszer gyermekbetegségei jóval a bevezetés után derültek ki. A CP/M ebből a szempontból kivételt jelent, működése teljesen biztonságos. Valójában miért van szükség a Commodore 64-es használatjának a DOS-on és a BASIC 2.0-n kívül másik operációs rendszerre?

Ezt a kérdést főként azok a Commodore használók teszik fel, akik még sohasem dolgoztak nagy rendszerekkel. A nagyszámítógépes rendszerek olyan mennyiségű szoftvertermékkel állnak a használók rendelkezésére, amelyről az egyszerű kívülállók (a Commodore-osok tábora) nem is álmodnak. A Commodore 64-esre viszonylag kevés önálló szoftvertermék készült, ezek számát hihetetlenül megnöveli a CP/M operációs rendszer használata. Az utóbbi időben ugyan megjelent néhány szövegszerkesztő, adatfeldolgozó stb. program a kereskedelmi forgalomban az addig kapható játékprogramokon kívül, de ez messze nem jelenti azt, hogy sikerült volna behozni a CP/M előnyét. A CP/M egyrészt tehát kiszélesíti a Commodore 64-esen alkalmazható szoftverek körét, másrészt ezek a programok jóval szélesebb piaccal rendelkeznek. A CP/M alatt fejlesztett rendszerek használóinak száma messze meghaladja a Commodore géppel rendelkezőkét. Meg kell azonban említenünk egy valóban létező gondot!

Mit jelent egy program CP/M alatti alkalmazhatósága? Az egyik CP/M vajon felismeri a másik alkalmazásával készült programot? Sajnos a

CP/M NEM MINDIG UGYANAZ A CP/M!

A számítógépek gyártói ugyanis gyakran meglepetést okoznak a használóiknak azzal, hogy bizonyos módosításokat végeznek a CP/M-en. Az ilyen változtatás többnyire azzal jár, hogy a rendszer kompatibilitása a többi géphez csak látszólagos. A Commodore 64 CP/M rendszerének is megvannak a maga sajátosságai. Így például, ebbe a rendszerbe az I/O byte-ot egyáltalán nem építették be, így a Commodore 64-es csak a 40 karakteres sorokból álló képernyőszerkesztésre képes, és a lemezformátum sem egyezik a többi gépek lemezformátumával. Ezekre a problémákra a későbbiek során még visszatérünk.

A CP/M néhány fontos tulajdonsága:

- 1) Csak olyan gépeken használható, amely legalább 48 kbyte szabad RAM területtel rendelkezik.
- 2) A szabad területnek a \$0100-as címtől kell kezdődnie.
- 3) A legtöbb program a 80 oszlopos 24 soros képernyőfelbontást használja.
- 4) A legtöbb CP/M szoftver csak 8"-(inch)-es lemezen működik.

Vegyük sorra ezeket a tulajdonságokat. A CP/M minimum 48 kbyte-os munkaterületet igényel. Nagyon sok mikrogép tárhelykapacitása bővíthető 48 kbyte-ra. A Commodore 64 már alapkiépítésben nagyobb kapacitással rendelkezik. A gép további sajátossága, a teljes ROM terület kikapcsolhatósága (l. „A 64 Belső felépítése” c. könyvet). A ROM kikapcsolásával a ROM területen tárolt rutinokat a CP/M alatt is használhatjuk. Ugyanezt más gépeken sehol, vagy csak nagyon körülményesen oldhatjuk meg. Van pl. olyan gép, amelynek a ROM területe a CP/M tárhelyosztása szempontjából olyan szerencsétlen helyen van (pl. 0000-tól), hogy ezen a gépen a CP/M operációs rendszert a megszokottól teljesen eltérő módon kell kialakítani. Azok a programozók, akik ilyen operációs rendszerrel dolgoznak, és pl. meg akarják keresni az egyes CP/M rutinokat, beleöszülnek a vállalkozásba. A kompatibilitás ilyen esetekben szóba sem jöhet. A Commodore 64-es szerencsére nem tartozik az ilyen tárhelyosztású gépek közé. A CP/M a Commodore 64-esen a megszokott helyre kerül (a \$0100 tárcímától kezdve).

Az első gondot a CP/M és CBM 64-es illeszkedésében a képernyő kijelzőmódja okozza. A CP/M, éppúgy mint a Word Star, a Data Star, és egyéb szoftverek, 80 karakteres kijelzést igényel, amire a CBM 64-es nem képes. Az illesztés mégsem megoldhatatlan! Erről bővebben a 6.3 fejezetben találunk Olvasóink.

Az utolsó problémát a lemez okozza. Mint azt említettük, a legtöbb alkalmazói szoftver 8"-es, a 1541-es lemezegység pedig 5 1/4"-es lemezekkel dolgozik. Egyetlen vigasz, hogy az utóbbi időben egyre több számítógépgyártó cég tér át az 5 1/4"-es lemezek használatára, így egyre több olyan szoftvert dolgoznak ki, amely a különböző lemezek közötti adatátvitelt biztosítja.

Hogyan épül fel a CP/M?

A CP/M operációs rendszer négy különböző egységből áll:

1) BIOS (Basic Input/Output System)

Ahogy a neve is mutatja, a BIOS biztosítja a rendszer és a külvilág kapcsolatát, azaz továbbítja például a nyomtató, a képernyő vagy a lemezegységnek szóló üzeneteket.

Az operációs rendszer egy BIOS-beli függvényhívással értesül az éppen folyamatban levő eseményekről.

2) BDOS (Basic Disk Operating System)

Ez a rész vezérli a lemezegységeket – azaz kezeli a lemez tartalmát, a tartalomjegyzéket és lebonyolítja a tényleges írási és olvasási utasításokat. A műveletek végrehajtása függvényhívással történik.

3) CCP (Console Command Processor)

Minden operációs rendszerrel valamilyen módon közölnünk kell a kívánságainkat. Az üzenetváltás eszköze a Commodore 64-esen a billentyűzet. A CCP továbbítja a billentyűzetről érkező üzeneteket az operációs rendszernek – ezt úgy kell elképzelni, mint egy telefonvonalas kapcsolatot az adó és a vevő között.

4) TPA (Transient Program Area)

A TPA az a szabad tárterület, ahol a rendszer a használó által írt programokat tárolja.

A BIOS, a BDOS, a CCP és a TPA elhelyezkedése a tárban:

| <i>Jel</i> | <i>Kezdőcím</i> |
|---------------------|-----------------|
| FDOS (BIOS + BDOS) | \$9C00 |
| CCP | \$9400 |
| TPA | \$0100 |
| Rendszerparaméterek | \$0000 |

AZ FDOS MŰVELETEI:

Műveletkód *Megjelölés*

BIOS

| | |
|----|---|
| 00 | Rendszer reset |
| 01 | ASCII karakter a terminálról |
| 02 | ASCII karakter a terminálra |
| 03 | Egy karakter a szalagolvasóról |
| 04 | Egy karakter a szalaglyukasztóra |
| 05 | Egy karakter a nyomtatóról |
| 06 | Egy karakter a nyomtatóra |
| 07 | Egy külső egység státuszának leolvasása |
| 08 | Státusz küldése egy külső egységhez |
| 09 | Karakterlánc a pufferből |
| 10 | Karakterlánc a pufferbe |
| 11 | A konzol státusz beolvasása |

BDOS

| | |
|----|--|
| 12 | Változatszám olvasása a CP/M-ből |
| 13 | Disk reset |
| 14 | Egységszámválasztás |
| 15 | Egy file megnyitása (OPEN) |
| 16 | Egy file lezárása (CLOSE) |
| 17 | Az első program megkeresése az FCB-ben |
| 18 | A következő program megkeresése az FCB-ben |
| 19 | Program törlése (DELETE) |
| 20 | Olvasás egy soros file-ból |
| 21 | Írás egy soros file-ba |
| 22 | Egy file létrehozása |
| 23 | Egy program nevének megváltoztatása (RENAME) |
| 24 | Az elérhető egységek megadása |
| 25 | Az aktuális egységszám beolvasása |
| 26 | A DMA cím beállítása |
| 27 | Egy cím beolvasása |
| 28 | Írásvédelem |
| 29 | Az írás/olvasás kapcsoló leolvasása |
| 30 | Az írás/olvasás kapcsoló beállítása |
| 31 | A lemezparaméterek leolvasása |
| 32 | Az ID írása/olvasása |

- 33 Olvasás egy közvetlen elérésű file-ból
- 34 Írás egy közvetlen elérésű file-ba
- 35 A program hosszának kiszámítása
- 36 A rekord címének beolvasása

A rendszer minden rutint egy sajátos „minta” alapján hív meg. Működésének megértéséhez szükség van a 8080-as processzor gépi nyelvének ismeretére. Mivel a CP/M-et ezzel a processzorral fejlesztették, és a Commodore 64-esbe egy Z80-as processzor van beépítve, ami „megérti” a 8080-as gépi nyelvet, a legtöbb CP/M adaptálás ehhez a nyelvhez igazodik. Ha valaki nem ismeri ezt a nyelvet, de mélyebben szeretne foglalkozni a CP/M-mel, annak azt javasoljuk, hogy ne csak egy CP/M kézikönyvet (pl.: CP/M Handbuch – RODNAY ZAKS) tanulmányozzon át, hanem egy Z80-as vagy 8080-as kézikönyvet is.

Hogyan végzi el a rendszer az egyes műveleteket?

Nézzünk erre egy példát. A Commodore CP/M változatszámának megállapításához az alábbi rutinok szükségesek:

```
MVI  C,12    ;12-es művelet
CALL 0005    ;ugrás a BDOS-ba
CPI   20H    ;$20 a CP/M 2.0 változatra utal
```

A C regiszterbe betöltődik a 12, mint a változatszámot meghatározó művelet azonosító kódja. Elágazás (a 0005-ös címre) előtt a C regiszterbe mindig bekerül a műveletkód. A műveletkód alapján a CP/M tudja, hogy a változatszámot kell meghatározni, és automatikusan a rendszer olyan részére ugrik, amely elvégzi az olvasást. Ezután a BDOS ismét visszatér a gépi kódú programra, és a változatszámot tárolja az akkumulátorban. Ha az akkumulátor tartalma \$20, arra következtethetünk, hogy a változatszám legalább 2.0.

Programozás közben ez az információ nagyon fontos, ugyanis a 2.0-nál alacsonyabb változatok nem ismerik pl. a közvetlen elérésű file-t.

A programot másik gépen csak akkor futtathatjuk, ha annak CP/M változatszáma legalább 2.0.

A CP/M művelethívásai alapvetően a regiszterek tartalmára épülnek. A legfontosabb a C regiszter, amely a mindenkor művelet kódját tartalmazza, és amelynek természetesen a BDOS/BIOS hívása előtt értéket kell adni. A kívánt művelet elvégzése után a szükséges információk a többi regiszterben állnak rendelkezésre. Abban az esetben, ha nem információt kapunk, hanem információt küldünk, természetesen a hívás másképpen történik. A C regiszter ekkor is a műveletkódot tartalmazza, de az egyéb regiszterek tartalmát ez esetben a hívás előtt kell beállítani.

Miért műveletkód?

Mint már említettük, a CP/M operációs rendszer előnye a rugalmasság, azaz általában a CP/M alatt megírt program minden olyan gépen futtatható, amelyen van CP/M. A rugalmasságot az biztosítja, hogy a BIOS/BDOS-t, ill. a CCP-t könnyen meg lehet változtatni a géptípusnak megfelelően.

A programozónak nem kell foglalkoznia az eltérésekkel, nyugodt lehet a program „szállíthatósága” felől, ugyanis az operációs rendszer a BIOS/BDOS hívásokat nem közvetlen címzés-sel, hanem a 0005-ös tárcímen keresztül bonyolítja le.

A CP/M egyes részeit bármikor megváltoztathatjuk anélkül, hogy az a rendszer működésének lényegét érintené. A CP/M felépítése hasonló a Commodore 64-es Kernai-ROM felépítéséhez, ahol a Kernal rutinok belépési pontjait egy ún. vektortáblázat tartalmazza. Ha az egyes rutinokat megváltoztatjuk, a régi programok ugyanúgy futnak, mint korábban, „észre sem veszik”, hogy közben az operációs rendszer megváltozott.

A CCP utasításai

A CCP a programozó és a CP/M közötti kapcsolatteremtés eszköze (interface). A CCP-ből tetszőleges programot futtathatunk, de használhatjuk a CCP saját utasításkészletét is.

1) DIR (Directory): Ez az utasítás megjeleníti képernyőn a lemez tartalomjegyzékét.

Az utasítás különböző változatai:

– DIR: az aktuális lemezegység teljes tartalomjegyzékének kiírása.

– DIR B: a B (1) lemezegységben található lemez teljes tartalomjegyzékének kiírása. A B (1) helyett A (0)-t is írhatunk, ekkor az utasítás a másik egységre vonatkozik.

– DIR (NEV.EXT): az utasítás hatására a rendszer megvizsgálja, hogy az adott nevű program szerepel-e a lemezen vagy sem.

A NEV tetszőleges, maximum nyolc karakterből álló, betűvel kezdődő és különleges karaktereket nem tartalmazó szöveg lehet.

Az EXT a programnév három betűs kibővítése, ami a program típusára utal. Pl. ha ez a bővítés COM – a program betöltés után azonnal indul, ha TXT, akkor szöveget tartalmazó file-ról van szó, a BAS – BASIC programra utal stb.

– DIR (*EXT): az összes olyan programnyelvet kijelzi, amelyek az adott típusba tartoznak.

Pl. a DIR*. COM utasítás az összes közvetlenül futó programot megkeresi a lemezen.

A CP/M kézikönyvben megtaláljuk a DIR utasítás összes opcióját.

Nézzünk erre egy példát:

```
A> DIR
A : MOVCPM COM : PIP COM
A : ED          COM : ASM COM
A : DDT         COM : LOAD COM
A : STAT        COM : SYSGEN COM
A : DUMP        COM : DUMP ASM
A : COPY        COM : CONFIG COM
```

2) ERA (ERASE): egy vagy több programot töröl a lemezről.

Itt is többféle opció létezik:

– ERA <NEV.EXT>

– ERA <*EXT>

3) REN (REName): a lemezen található valamely file vagy program átnevezése. Az utasítás csak egy formában használható:

– REN <új név> = <régi név>

4) TYPE: csak a szöveg file-ok esetén használható. Kiírja a képernyőre a file tartalmát. Formátuma:

– TYPE <file-név.EXT>

A típus TXT, PRN vagy ezekhez hasonló lehet.

5) SAVE: ez az utasítás első látásra kicsit bonyolultnak tűnhet. Elsősorban a DDT-vel módosított programok, ill. a módosított CP/M változatok tárolására használjuk. Formátuma:

– SAVE <lapok száma> <név.EXT>

A lapok száma a 256 byte-os egységek számát jelenti. A következő parancs:

SAVE 50 TESZT.COM

a tár \$0100-as címétől kezdve (a TPA kezdőcíme) a \$32FF-ig terjedő tárterület tartalmát TESZT.COM címen a lemezre menti. A hossz: 50•256 byte.

6) USER: ezzel az utasítással feloszthatjuk a lemezen tárolt programokat az egyes használok között, azaz megakadályozhatjuk, hogy a programokhoz bárki hozzáférjen. Ennek természetesen a Commodore 64-esen nincs sok jelentősége.

Az utasítás formátuma:

– USER a használok kódja

A használok kódja egy egész szám 0-tól 15-ig.

Az adott kóddal azonosított használok csak a számára kijelölt programok tartalomjegyzékét érheti el. A használok kód alapértelmezése 0.

A fenti utasítások többségét minden CP/M változat megérti, kivéve a USER utasítást, amely a 2.0 változat kifejlesztése után került a rendszerbe.

A következő fejezetben ismertetjük a CP/M rendszer programjait: a PIP, az ED, a DDT és a STAT programokat. Ezek minden CP/M rendszerlemezen megtalálhatóak, és nem parancsok, hanem programok, amelyek betöltés után bővítik a CP/M rendszer utasításkészletét.

6.2 A CP/M rendszer programjai

A következő programok tartoznak a CP/M standard változatához:

- STAT.COM Általános rendszerinformációkat (pl. egy lemez szabad kapacitása, egység kijelölések stb.) szolgáltató program.
- ASM.COM A 8080-as processzor assembly nyelvének assemblere. (A gépi kódhoz legközelebb álló mnemonikokkal készült programnyelv fordítója.)
- LOAD.COM Megszerkeszti a lefordított programok futtatható változatát (a lemezen a futtatható modulok neve után a .HEX rövidítés szerepel).
- DUMP.COM A .COM típusú programokat olvasható formában (hexadecimálisan) megjeleníti a képernyőn.
- PIP.COM Lebonyolítja a különböző külső egységek közötti adatforgalmat.
- ED.COM A CP/M szövegszerkesztője. Megkönnyíti a szövegek, assembler nyelvű programok stb. tárolását és szerkesztését.
- SYSGEN.COM Míg a PIP programmal csak file-okat tudunk másolni, a SYSGEN a lemezt sávonként eléri, másolja, ill. létrehoz új BIOS sávokat. (l. 6.3)
- MOVCPM.COM Biztosítja a CP/M standard változatának illeszkedését az egyes géptípusokhoz.
- SUBMIT.COM Az adatbevitel során gyakran előfordul, hogy valamilyen hiba folytán az állomány egy része elvész, és az adatbevitelt meg kell ismételni. A SUBMIT programmal biztonsági input-file-okat hozhatunk létre, melyek feltöltése az előzetesen rögzített paraméterektől függő időpontokban kezdődhet.
- XSUB.COM Ez a program megkönnyíti a gyakran ismétlődő utasítások begépelését. Csak a SUBMIT programmal együtt használható. A SUBMIT futása közben dolgozhatunk a billentyűzeten.

Az egyes programok kimerítő ismertetése meghaladja e könyv kereteit, az érdeklődők további ismereteket találnak a CP/M kézikönyvben.

Az alábbiakban a három legfontosabb programról ejtünk még néhány szót.

STAT

A rendszer egyik legfontosabb programja a STAT program, amelyet STATUS programnak is nevezhetünk. Ez a program számos információt szolgáltat a rendszer pillanatnyi állapotáról. Nemcsak a lemez szabad kapacitását, az egyes programok hosszát, hanem többek között az író/olvasó fej pillanatnyi állapotát (írás- vagy olvasási fázis) is le tudjuk kérdezni, sőt az erre vonatkozó kapcsoló beállításával meg is tudjuk változtatni ezt az állapotot. Lekérdezhetjük az egyes hozzárendeléseket is. Ezen a ponton meg kell említeni a Commodore 64-es CP/M változat egy korlátját: minthogy IOBYTE nincs beépítve, a hozzárendeléseket a STAT programmal sem tudjuk megváltoztatni, ami azonban a munka során általában nem zavaró.

Nézzünk egy példát a STAT használatára:

A> STAT VAL:

TEMP R/O DISK: D:=R/O

SET INDICATOR: D:FILENEV.TYP \$R/O \$R/W \$SYS \$DIR

DISK STATUS : DSK : D:DSK:
USER STATUS : USR:
IOBYTE ASSIGN:

CON: = TTY : CRT : BAT : UC1:
RDR: = TTY : PRT : UR1 : UR2:
PUN: = TTY : PTP : UP1 : UP2:
LST: = TTY : CRT : LPT : UL1:

A fentiekből látható, hogy mely értékeket és milyen módon változtathatunk meg (legalábbis elméletileg), de mint már említettük, az IOBYTE ASSIGN értékek a C 64-esen nem változtathatók meg.

Ha például a lemezt írástól védeni akarjuk, akkor elegendő a következőket beírni: STAT A:=R/O. Ez az írásvédelem addig marad érvényben, amíg a gépet ki nem kapcsoljuk. A CP/M egység kiutalásához gépeljük be a következő utasítást: STAT A:DEV:
Az eredmény a C-64-esen a következőképpen néz ki:

A > STAT DEV:

CON: IS TTY:
RDR: IS TTY:
PUN: IS TTY:
LST: IS TTY:

Az említettek miatt a STAT LST:=LPT: módosításnak a fenti hozzárendelésekre nincs semmi hatása.

A lemez paramétereit a következő utasítással kérdezzük le:

A > STAT DSK:

A : DRIVE CHARACTERISTICS (lemezegység azonosító)
1088 : 128 BYTE RECORD CAPACITY (szabad terület rekordban)
136 : KILOBYTE DRIVE CAPACITY (szabad terület kbyte-ban)
64 : 32 BYTE DIRECTORY ENTRIES (file bejegyzések száma)
128 : REOCRDS/EXTENT
8 : RECORDS/BLOCK
34 : SECTORS/BLOCK
2 : RESERVED TRACKS

PIP

Általános file-másoló program. A PIP programmal az egységek között másoláson kívül a nyomtatóra is listázhatunk szöveges file-okat. Rendkívül előnyös, hogy a nyomtatás képét megszerkeszthetjük, azaz a nyomtatott szöveget pl. sorszámozhatjuk, tabulálhatjuk stb.
A következő utasítás a teljes lemez tartalmát átmásolja az A egységről a B egységre: PIP B:=A:*. A PIP LIST=DUMP ASM utasítással pedig kiírathatjuk nyomtatóra a DUMP.ASM szövegfájl-t, feltéve, hogy a nyomtatót előzetesen csatlakoztattuk a géphez.

ED

Az ED szövegszerkesztővel tetszőleges szövegeket, gépkódú programokat, egyéb nyelveken írt (FORTRAN, COBOL stb.) lefordítandó programokat gépelhetünk be és tárolhatunk a lemezen. Az ED kezeléséhez bizonyos gyakorlatra van szükség, és bár lehet, hogy a Commodore használóknak ez eleinte kissé bonyolultnak tűnik, gazdag szolgáltatásai megérik a fáradságot.

Lássunk ismét egy példát:

```
A>ED TESZT.TXT
NEW FILE
*|
EZ A SZOVEG ELSO SORA
EZ A MASODIK
<CTRL> -Z (Nyomja le egyszerre a CTRL és a Z billentyűt)
*E
```

A fenti néhány utasítással egy szöveg első két sorát lemezen tároltuk a TESZT.TXT nevű file-ban. A csillag az ED program közvetlen parancsa. Az E hatására a szerkesztés befejeződik, a program tárolja a szöveget lemezen, majd a CP/M az A> várakozó üzenettel visszajelentkezik. A tárolt szöveget a TYPE TESZT.TXT utasítással bármikor megjeleníthetjük képernyőn. A szövegszerkesztés utasításkészletét (módosítás, törlés, beszúrás, csere stb.) a CP/M kézikönyv részletesen tartalmazza.

6.3 Standard CP/M szoftverek a CBM 64-esen

Mire kell figyelnünk, amikor egy CP/M szoftvert akarunk illeszteni a Commodore 64-es hardveréhez?

Először is arra, hogy a legtöbb CP/M szoftver 80 oszlopos kiírással dolgozik, szemben a CBM 64-es 40 oszlopos kijelzésével, tehát feltétlenül be kell szerezni egy 80 karakteres kártyát. Másrészt ahhoz, hogy a CP/M előnyeit kihasználhassuk, célszerű nagy és gyors CBM lemezegységet beszereznünk. Magát a CP/M-et a MOVCPM és SYSGEN programok illesztik az egyes géptípusokhoz. A MOVCPM program szervezi az operációs rendszer szükséges tárkiosztását mégpedig úgy, hogy a teljes tárkapacitás kihasználható legyen. A CP/M rendszer indításakor a gép üzenete: 44 K CP/M.

A SYSGEN (rendszergenerálás) programmal a CP/M-et átmásolhatjuk a saját lemezünkre.

6.4. A Z80-as processzor tárkiosztása

A Z80-as processzor a CP/M kártyán a Commodore 64-es teljes RAM területét (64 kbyte-ot) meg tudja címezni. A Z80-as processzor startcímként (reset) a 0-s címet használja, szemben a 6510-essel, melyen ez a cím a processzor port címe. Emiatt a tár címzése másként történik a 6510-es processzoron, és másként a Z80-as processzoron keresztül. A címzés módok közötti eltérést, egy, a CP/M kártyára beépített hardver kezeli. Ha a Z80-as a 0-s tárcímre hivatkozik, a hardver a címvezetéken a 6510-eshez a \$1000-es címet közvetíti. Ahhoz tehát, hogy a 6510-es tárcíméből Z80-as tárcímet kapjunk, az előbbiből \$1000-et le kell vonnunk, vagy hozzáadunk \$F000-t úgy, hogy az átvitt figyelmen kívül hagyjuk. Így a 6510-es 0-\$1000-ig terjedő 4 kbyte területét a Z80-as címtartományának végére helyezzük át. Ezt a területet a 6510-es nulláslapja, a verem, a munkaterület, ill. a videoram foglalja le. A maradék 2 kbyte-os (\$800-tól \$FFF-ig) területen bonyolítja le a CP/M a két processzor közötti adatforgalmat, ill. ezt a területet használják a 6510-es I/O rutinjai programterületként.

A Z80-as minden I/O műveletet továbbít a 6510-eshez.

A következő ábra grafikusán szemlélteti az elmondottakat.

A legfontosabb közös tárcímek:

| Cím | 6510 | Z80 | |
|---------|--------|---------|----------------------------------|
| HSTBUF | \$0800 | 0F800H | 256 byte-os lemezpuffer |
| CMD | \$0900 | 0F900H | A 6510-es utasítás regisztere |
| DATA | \$0901 | 0F901H | Adatregiszter |
| SECTOR | \$0902 | 0F902H | Szektorregiszter |
| TRACK | \$0903 | 0F903H | Sávregiszter |
| DISKNO | \$0904 | 0F904H | Az egységszám regisztere |
| KEYCHAR | \$0905 | 0F905H | A lenyomott billentyű regisztere |
| MODESW | \$DE00 | 0FCE00H | Kapcsoló: 6510/Z80 |
| IOTYPE | \$0CFF | 0DCFFH | I/O konfiguráció |

6510

Z80

6510

\$FFFF

0EFFFH

0FFFH

\$0FFF

6510
operációs
rendszer

BIOS65

\$D000

0C000H

48K

\$C000

0B000H

44K

0F900H

\$0900

lemez-
puffer

0FB00H

\$0800

CP/M

videoram

0F400H

\$0400

\$1000

00000H

6510-Z80

6510
munka-
terület

\$0000

0FFFFH

0F000H

\$0000

6.5 Lemezkezelés CP/M alatt

A lemez koncentrikus sávokra, az egyes sávok pedig szektorokra vannak felosztva. A VC 1541-es lemez felosztása:

| Sáv | Szektor |
|-------|---------|
| 1–17 | 0–20 |
| 18–24 | 0–18 |
| 25–30 | 0–17 |
| 31–35 | 0–16 |

Az összesen 683 szektor (blokk) közül a 18-as sáv szektorain helyezkedik el a lemez tartalomjegyzéke, a maradék 664 blokk pedig programok és adatok tárolására használható.

A CP/M az első két sávot az operációs rendszer számára tartja fenn, a 18-as sávot felszabadítja, tehát CP/M rendszerrel ezen a területen is tárolhatunk programokat és adatokat. A CP/M nem dolgozik változó szektorszámmal, minden sávon egyöntetűen 17 szektort kezel, 0–16-ig számozva. Rendelkezésünkre áll tehát 32 sáv, ezek mindegyikén 17 szektor, ez összesen 574 db 256 byte-os blokk, azaz 143 kbyte. Ebből valamennyit még lefoglal a CP/M tartalomjegyzéke (max. 64 file-bejegyzés, egyenként 32 byte, ez összesen max. 2 kbyte). A tartalomjegyzék a BIOS-ban (Basic Input/Output System) helyezkedik el, az ún. Disk Parameter Blockban, amelyben a használó a paramétereket a saját lemezkapacitásának megfelelően alakíthatja. A BIOS-ban található többek között a 18-as sáv tartalma (a Commodore 64-es tartalomjegyzéke).

Az operációs rendszer által lefoglalt első két sáv felosztása a következő:

Az 1-es sáv 0-ás szektorra tartalmazza a CPM betöltőprogramot. Az 1-es sáv 1-es szektorától az 5-ös szektorig helyezkedik el a BIOS 65, amely a 6510-es I/O rutinból, ill. a CP/M hidegstart betöltőjéből áll. A CPM a BIOS 65-öt a \$0A00 – \$0EFF közötti tárterületre tölti be. Az ezt követő blokkok tartalma \$0E00-tól \$0EFF-ig a \$1000–\$10FF tárterületre töltődik. Ez a Z80-as 0-ás címe, ahová a hidegstart betöltő kerül az átvitel során. Végül, míg a Z80-as processzor be-, addig a 6510-es kikapcsol. A Z80-as ekkor elindítja a 0-ás címtől elhelyezett programot, amely betölti a CP/M rendszert a lemezről. A CCP (Command Control Processor) és a BDOS (Basic Disk Operating System) az 1-es és 2-es sávon 22 szektort foglal le, 1-től 6-ig, ill. 2-től 10-ig. Ezek tartalmát minden melegstart (CTRL-C) újra betölti, és minden betöltött szektor után megjelenik egy csillag a képernyőn. A BIOS-t, amely 5 szektort foglal el a 2-es sávon (11-től 16-ig), csak a hidegstart tölti a tárbá. Végül a tartalomjegyzék a 3-as sáv 0-tól 7-ig terjedő szektorain található.

A CP/M COPY segédprogramja (utility) csak kiválasztott szektorokat másol, pl. a 'system track only' feltétel megadásakor az 1-es, 2-es szektorokat, ill. a 18-as és 3-as sávokat.

Ha munka közben át akarunk kapcsolni egy másik lemezegységre, pl. az IEC Buson keresztül, az új egység sáv- és szektorfelosztását közölnünk kell a rendszerrel. A 4040-es egységnél semmilyen intézkedésre nincs szükség, hiszen ez teljesen kompatibilis a 1541-essel. A 8050-es és a 8250-es nagyobb tárhelykapacitását csak akkor tudjuk kihasználni, ha a változásról a BIOS-t értesítjük. Erre szolgál a Disk Parameter Block. A DPB-ben található a sávonkénti szektorszám (8050-esnél 23), a lemezkapacitás stb. A 18-as sáv helyett ezeknél az egységeknél a 38-as, ill. 39-es sávokat kell kihagyni a tartalomjegyzék számára.

A megfelelő változtatásokat a COPY programmal végezhetjük el.

6.6 Együttműködés a 6510-es és a Z80-as processzorok között

Miközben a Commodore 64-esen a CP/M rendszerrel dolgozunk, a két mikroprocesszor megosztja a feladatokat. Míg a Z80-as kiszolgálja a CP/M-et, a 6510-es végrehajt egy I/O műveletet, hiszen a Commodore 64-es operációs rendszere az input/output rutinokat rendelkezésre bocsátja. A Z80-as a következő utasításokat továbbítja a 6510-eshez:

| <i>Utasításkód</i> | <i>Művelet</i> |
|--------------------|--|
| 0 | Egy szektor beolvasása lemezzől |
| 1 | Egy szektor felírása lemezre |
| 2 | A billentyűzet lekérdezése |
| 3 | Egy karakter kiírása képernyőre |
| 4 | A lemezállapot (status) betöltése |
| 5 | Egy karakter felírása lemezre |
| 6 | A lemez formálása |
| 7-9 | További bővítésre fenntartva (pl. soros I/O) |

A két processzor csak felváltva működhet. Az átkapcsolást a 6510-esen a \$DF00, a Z80-ason a 0CE00H címek tartalma jelzi. Ha a Z80-as a 6510-eshez egy I/O műveletet továbbít, átadja a megfelelő utasításkódot (a fenti táblázatnak megfelelően), a közös tárcímeken keresztül közli az utasítás elvégzéséhez szükséges paramétereket, majd a 0CE00H címre 1-et írva, önmagát kikapcsolja, ezáltal bekapcsolva a 6510-est. A 6510-es végrehajtja a műveletet, majd a \$DE00-s címre 0-t írva, visszaadja a vezérlést a Z80-asnak, amely a hívás pontjától folytatja tovább saját programja futtatását.

Hardver okok miatt minden visszakapcsolás után egy NOP utasításnak kell következnie.

A paraméterek:

| 6510 | Z80 | Jelentés |
|-------|--------|---|
| \$900 | 0F900H | Utasításkód |
| \$901 | 0F901H | Adatok az I/O-hoz |
| \$902 | 0F902H | Szektorszám |
| \$903 | 0F903H | Sávszám |
| \$904 | 0F904H | Lemez kód |
| \$905 | 0F905H | Billentyűszám a billentyűzet lekérdezésekor |

A \$800-\$8FF, ill. 0F800H-0F8FFH közötti tárcímek I/O pufferként az aktuális (írt/olvasott) szektort tartalmazzák.

A szektorok írása/olvasása közvetlen hozzáférésű Block-Write/Block-Read utasításokkal történik.

A billentyűzet lekérdezése során a billentyűszám bekerül a megfelelő tárcímre. A billentyűszámnak megfelelő ASCII kódot a rendszer a BIOS-ból, a \$D00-\$DFF, ill. 0FD00H-0FDFFH tárcímeken található táblázatból határozza meg. A \$C00, ill. 0FC00H tárcímeken van annak a szövegnek a címe, amelyet a funkcióbillentyűkhöz rendeltünk, maguk a szövegek pedig a \$C10, ill. \$FC10H címen kezdődnek. A hozzárendeléseket a CONFIG programmal tetszőlegesen megváltoztathatjuk.

6.7 A CP/M BIOS listájának dokumentációja

A következő oldalakon a Commodore 64-es CP/M BASIC I/O System (BIOS) listáját közöljük teljes részletességgel, magyarázatokkal ellátva. A BIOS négy részből áll. Az első rész tartalmazza a CPM programot, amelyet a LOAD "CPM", 8 utasítással tölthetünk be a \$800-tól \$8FF-ig terjedő tárterületre. A CPM program betölti a BIOS 65-öt \$A00-tól \$DFF-ig. A BIOS 65 a BIOS-nak a 6510-es processzor által végrehajtott része. A harmadik rész az ún. BOOT program, amely a CP/M hidegstartjánál a negyedik részt, a tulajdonképpeni Z80 – BIOS-t betölti.

6.7.1 A BIOS 65 betöltőprogramja

```
0 REM **** F54. ****
1 :
2 :
100 : *****; A CP/M BEHIVOJA A C-64-EN
110 : 0B0F 7B SEI
120 : 0B10 20 E7 FF JSR $FFE7 ;AZ I/O CSATORNAK LEZARASA
130 : 0B13 A3 0F LDA #$0F ;15-OS LOGIKAI FILE SZAM
140 : 0B15 A2 08 LDX #$08 ;8-AS EGYSEG SZAM
150 : 0B17 AC 0F LDY #$0F ;15-OS MASODLAGOS CIM
160 : 0B19 20 BA FF JSR $FFBA ;FILE PARAMETEREK BEALLITASA
170 : 0B1C A9 00 LDA #$00 ;NINCS FILE NEV
180 : 0B1E 20 BD FF JSR $FFBD ;PARAMETEREK A FILE-NEVHEZ
190 : 0B21 20 C0 FF JSR $FFC0 ;HIBACSATORNA
200 : 0B24 A9 02 LDA #$02 ;2-ES LOGIKAI FILE SZAM
210 : 0B26 A2 08 LDX #$08 ;8-AS EGYSEG SZAM
220 : 0B28 AC 02 LDY #$02 ;2-ES MASODLAGOS CIM
230 : 0B2A 20 BA FF JSR $FFBA ;FILE-PARAMETEREK BEALLITASA
240 : 0B2D A9 01 LDA #$01 ;A FILE-NEV HOSSZA 1
250 : 0B2F A4 B5 LDX #$B5
260 : 0B31 A4 08 LDY #$08 ;MUTATO A "#"-RA
270 : 0B33 20 BD FF JSR $FFBD ;PARAMETEREK A FILE-NEVHEZ
280 : 0B36 20 C0 FF JSR $FFC0 ;KOZVETLEN ELERESU CSATORNA
290 : 0B39 A9 05 LDA #$05
300 : 0B3B 0D B6 0B STA $0B6 ;SZAMLALO 5 SZEKTOR BETOLTESEHEZ
310 : 0B3E A2 0F LDX #$0F ;15
320 : 0B40 20 C9 FF JSR $FFC9 ;CHKOUT, KIIRAS A 15-OS CSATORNAN
330 : 0B43 64 00 LDX #$00
340 : 0B45 A0 0B LDY #$0B ;12 KARAKTER
350 : 0B47 FD AA 08 LDA $0BAA,X ;"U1:2 0 1 S" 1-ES SAV 1-5 SZEKTORA
360 : 0B4A 20 D2 FF JSR $FFD2 ;KIIRAS
370 : 0B4D E9 INX
380 : 0B4E 68 DEY
390 : 0B4F D0 F6 BNE $0B47
400 : 0B51 7C CC FF JSR $FFCC ;CLRCH, KIVITEL ALAPERTELMEZES SZERINT
410 : 0B54 62 02 LDX #$02 ;2
420 : 0B56 20 C6 FF JSR $FFC6 ;CHKIN, BEOLVASAS A 2-ES CSATORNAROL
430 : 0B59 64 00 LDX #$00
440 : 0B5B 20 CF FF JSR $FFCF ;1 KARAKTER BETOLTESE
450 : 0B5E 70 0A STA $0A00,X ;ES TAROLASA
460 : 0B61 E9 INX
470 : 0B62 D0 F7 BNE $0B5B ;A 256. KARAKTER
480 : 0B64 7C CC FF JSR $FFCC ;CLRCH, BEOLVASAS ALAPERTELMEZES SZERINT
490 : 0B67 EE B3 08 INC $0BB3 ;SZEKTORSZAM NOVELESE
500 : 0B6A FF 60 08 INC $0B60 ;A TARCIM NOVELESE EGY LAPPAL
510 : 0B6D 64 B6 08 DEC $0BB6 ;5 SZEKTOR BEOLVASVA
520 : 0B70 F0 CC BNE $0B3E ;NEM, TOVABBI OLVASAS
530 : 0B72 64 00 LDX #$00
540 : 0B74 D0 0E LDA $0E00,X ;A Z80-AS BOOT PROGRAMJA
```

```

550 : 0877 9D 00 10 STA #1000,X ;MASOLAS A 0000H Z80-AS CIM SZERINT
560 : 087A E8 INX
570 : 087B D0 F7 BNE #0874
580 : 087D 20 E7 FF JSR #FFE7 ;CLALL, I/O CSATORNAK LEZARASA
590 : 0880 A9 36 LDA #36
600 : 0882 85 01 STA #01 ;BASIC ROM KIKAPCSOLASA
610 : 088A A9 09 LDA #09 ;CHR$(9)
620 : 0886 20 D2 FF JSR #FFD2 ;AZ ATKAPCSOLAS ENGEDELYEZETT !
630 : 0889 A9 0E LDA #0E ;CHR$(14)
640 : 088B 20 D2 FF JSR #FFD2 ;SZOVEG-MOD BEALLITASA
650 : 088E A9 08 LDA #08 ;CHR$(8)
660 : 0890 20 D2 FF JSR #FFD2 ;AZ ATKAPCSOLAS LETILTVA !
670 : 0893 A9 93 LDA #93 ;CHR$(147)
680 : 0895 20 D2 FF JSR #FFD2 ;KEPERNYOTORLES
690 : 0898 A9 0D LDA #0D ;CHR$(13)
700 : 089A 20 D2 FF JSR #FFD2 ;UJ SOR
710 : 089D A9 FF LDA #FF
720 : 089F 8D 00 09 STA #0900 ;UTASITASREGISZTER
730 : 08A2 A9 28 LDA #28 ;BILLENTYUSZAM
740 : 08A4 8D 05 09 STA #0905 ;TAROLASA
750 : 08A7 4C 00 0A JMP #0A00 ;VISSZA A Z80-6510 FOCIKLUSHOZ !
760 : 08AA 55 31 3A 32 20 30 20 31 20 31 0D "U1:2 0 1 1" ;BLOCK-READ UTASITAS
770 : 08B5 23 ; "0" FILE-NEV KOZVETLEN ELERESEHEZ
KESZ .

```

6.7.2 A BIOS 65

```

0 REM **** F55. ****
1 :
2 :
100 : *****;FOCIKLUS Z80-6510
110 : 0A00 A9 00 LDA #00
120 : 0A02 8D 00 DE STA #DE00 ;A 6510 KIKAPCSOLASA
130 : 0A05 EA NOP
140 : 0A06 20 0C 0A JSR #0A0C ;A Z80-AS RUTIN
150 : 0A09 4C 00 0A JMP #0A00 ;VISSZA A CIKLUSHOZ
160 :
170 : *****;A RUTINOK SZELEKTALASA
180 : 0A0C AD 00 09 LDA #0900 ;UTASITASKOD
190 : 0A0F C9 FF CMP #FF ;A Z80 NEM AKTIV
200 : 0A11 D0 03 BNE #0A16
210 : 0A13 6C FC FF JMP ($FFFC) ;A C64 RESET-JE
220 : 0A16 C9 0A CMP #0A ;AZ UTASITASKOD NAGYOBB V. EGYENLO MINT 10
230 : 0A18 90 01 BCC #0A1B ;NEM
240 : 0A1A 60 RTS ;KESZ
250 : 0A1B D8 CLD
260 : 0A1C 18 CLC
270 : 0A1D 6D 00 09 ADC #0900 ;UTASITASKOD * 2
280 : 0A20 69 28 ADC #28 ;PLUSZ #28
290 : 0A22 8D 26 0A STA #0A26 ;INDIREKT CIMZESMOD
300 : 0A25 6C 28 0A JMP ($0A2B) ;AZ UTASITAS VEGREHAJTASA
310 :
320 : *****;A 10 MUVELET UGRASIT TABLAZATA
330 : 0A28 3F 0A 0. UTASITAS ;SZEKTOR OLVASASA
340 : 0A2A 3F 0A 1. UTASITAS ;SZEKTOR IRASA
350 : 0A2C 86 0A 2. UTASITAS ;KLAVIATURA LEKERDEZESE
360 : 0A2E 8F 0A 3. UTASITAS ;KIIRAS KEPERNYORE
370 : 0A30 99 0A 4. UTASITAS ;NYOMTATO STATUSZ BEOLVASASA
380 : 0A32 9F 0A 5. UTASITAS ;KIIRAS NYOMTATORA
390 : 0A34 49 0B 6. UTASITAS ;LEMEZFORMALAS
400 : 0A36 00 0E 7. UTASITAS ;AUX1, UGRAS A #E00-RA
410 : 0A38 00 0F 8. UTASITAS ;AUX2, UGRAS A #F00-RA
420 : 0A3A 3C 0A 9. UTASITAS ;INDIREKT, UGRAS A ($906)-RA
430 :
440 : *****;A 9-ES UTASITAS, INDIREKT

```

```

450 : 0A3C 6C 06 09 JMP ($0906) ;INDIREKT UGRAS A ($906)-RA
460 :
470 : *****;A 0-AS UTASITAS, READ-SZEKTOR
480 : 0A3F A9 31 LDA #$31 ;"1", "U1" UTASITAS A LEMEZHEZ
490 : 0A41 20 F2 0A JSR $0AF2 ;BLOCK-READ
500 : 0A44 20 DE 0B JSR $0BDE ;LEOLVASAS A KOZVETLEN ELERESU CSATORNAROL
510 : 0A47 A2 00 LDX #$00
520 : 0A49 20 CF FF JSR $FFCF ;EGY KARAKTER BEOLVASASA,
530 : 0A4C 9D 00 08 STA $0B00,X ;TAROLASA A PUFFERBEN
540 : 0A4F E8 INX
550 : 0A50 D0 F7 BNE $0A49 ;A 256.KARAKTER
560 : 0A52 F0 60 BEQ $0AB4 ;CLRCH
570 : 0A54 20 97 0B JSR $0B97 ;A LEMEZ INICIALIZALASA
580 :
590 : *****;AZ 1-ES UTASITAS, WRITE-SZEKTOR
600 : 0A57 20 F4 0B JSR $0BF4 ;KIIRAS A LEMEZ PARANCSCSATORNAJARA
610 : 0A5A A0 08 LDY #$08 ;8
620 : 0A5C BD 80 0B LDA $0B80,X ;MUTATO BEALLITASA A SZEKTORKEZDETRE, ES
630 : 0A5F 20 D2 FF JSR $FFD2 ;KIIRASA
640 : 0A62 E8 INX
650 : 0A63 88 DEY
660 : 0A64 D0 F6 BNE $0A5C
670 : 0A66 20 CC FF JSR $FFCC ;CLRCH
680 : 0A69 20 CE 0B JSR $0BCE ;A HIBACSATORNA LEOLVASASA
690 : 0A6C D0 E6 BNE $0A54 ;HIBA, INICIALIZALAS, ISMETLES
700 : 0A6E 20 CC FF JSR $FFCC ;CLRCH
710 : 0A71 20 E9 0B JSR $0BE9 ;KIVITEL A 2-ES CSATORNARA
720 : 0A74 A2 00 LDX #$00
730 : 0A76 BD 00 0B LDA $0B00,X ;EGY KARAKTER A PUFFERBOL
740 : 0A79 20 D2 FF JSR $FFD2 ;A LEMEZHEZ
750 : 0A7C E8 INX
760 : 0A7D D0 F7 BNE $0A76 ;A 256. KARAKTER
770 : 0A7F 20 CC FF JSR $FFCC ;CLRCH
780 : 0A82 A9 32 LDA #$32 ;"2", "U2" BLOCK WRITE
790 : 0A84 D0 6C BNE $0AF2 ;BLOCK WRITE UTASITAS A LEMEZHEZ
800 :
810 : *****;A 2-ES UTASITAS, KLAVIATURA LEKERDEZESE
820 : 0A86 20 9F FF JSR $FF9F ;A KLAVIATURA LEKERDEZESE
830 : 0A89 A5 C5 LDA $C5 ;AZ OLVASOTT KARAKTER MATRIX SZAMANAK
840 : 0A8B 8D 05 09 STA $0905 ;TOVABITASA A Z80-HOZ
850 : 0A8E 60 RTS
860 :
870 : *****;A 3-AS UTASITAS, KIIRAS KEPERNYORE
880 : 0A8F A9 00 LDA #$00
890 : 0A91 85 D4 STA $D4 ;A FELSOVESSZO TORLESE
900 : 0A93 AD 01 09 LDA $0901 ;EGY KARAKTER BEOLVASASA,
910 : 0A96 4C D2 FF JMP $FFD2 ;ES KIIRASA A KEPERNYORE
920 :
930 : *****;A 4-ES UTASITAS, NYOMTATO STATUSZ BETOLTESE
940 : 0A99 A9 00 LDA #$00
950 : 0A9B BD 01 09 STA $0901 ;A STATUSZ 0
960 : 0A9E 60 RTS
970 :
980 : *****;AZ 5-OS UTASITAS, KIVITEL NYOMTATORA
990 : 0A9F AD 01 09 LDA $0901 ;A KIIRANDO KARAKTER
1000 : 0AA2 C9 0A CMP #$0A ;SOREMELES
1010 : 0AA4 D0 01 BNE $0AA7 ;NEM
1020 : 0AA6 60 RTS ;IGEN, UJ SOR
1030 : 0AA7 A2 04 LDX #$04 ;A LOGIKAI FILE-SZAM 4
1040 : 0AA9 20 C9 FF JSR $FFC9 ;CKOUT, KIIRAS A NYOMTATORA
1050 : 0AAC B0 09 BCS $0AB7 ;HIBA
1060 : 0AAE AD 01 09 LDA $0901 ;A KIIRANDO KARAKTER
1070 : 0AB1 20 D2 FF JSR $FFD2 ;ELKULDESE A NYOMTATHOZ
1080 : 0AB4 4C 0C JMP $FFCC ;CLRCH, A KIIRAS ALAPERTELMEZES SZERINT
1090 : 0AB7 C9 03 CMP #$03 ;FILE NOT OPEN
1100 : 0AB9 D0 05 BNE $0AC0 ;NEM, MAS HIBA
1110 : 0ABB 20 C6 0A JSR $0AC6 ;A NYOMTATO NYITASA

```

```

1120 : 0ABE 90 DF BCC $0A9F ;NEM VOLT HIBA, KARAKTER KIIRAS
1130 : 0AC0 A9 FF LDA #$FF ;A HIBA KODJAT A
1140 : 0AC2 BD 01 09 STA $0901 ;A ZB0-NAK ATADNI
1150 : 0AC5 60 RTS
1160 : 0AC6 A0 07 LDY #$07 ;A 7-ES MASODLAGOS CIM A VC. NYOMTATOHOZ
1170 : 0ACB 20 DE 0A JSR $0ADE ;A NYOMTATO LEZARASA, UJ ADATHALMAZ NYITAS
1180 : 0ACB AD FF 0C LDA $0CFF ;I/O-TIPUS (NYOMTATO TIPUS)
1190 : 0ACE 29 02 AND #$02
1200 : 0AD0 F0 F3 BEQ $0AC5 ;1515-OS NYOMTATO
1210 : 0AD2 A2 04 LDX #$04 ;LOGIKAI FILE-SZAM
1220 : 0AD4 20 C9 FF JSR $FFC9 ;CHKOUT, KIIRAS A NYOMTATORA
1230 : 0AD7 A9 0D LDA #$0D ;KOCSEI VISSZA A
1240 : 0AD9 20 D2 FF JSR $FFD2 ;NYOMTATON
1250 : 0ADC A0 00 LDY #$00 ;0-AS MASODLAGOS CIM
1260 : 0ADE A9 04 LDA #$04 ;A LOGIKAI FILE-SZAM 4
1270 : 0AE0 20 C3 FF JSR $FFC3 ;CLOSE
1280 : 0AE3 A9 04 LDA #$04 ;A LOGIKAI FILE-SZAM 4
1290 : 0AE5 A2 04 LDX #$04 ;A KESZULEKSZAM 4
1300 : 0AE7 20 BA FF JSR $FFBA ;A PARAMETER BEALLITASA
1310 : 0AEA A9 00 LDA #$00 ;NINCS FILE-NEV
1320 : 0AEC 20 BD FF JSR $FFBD ;PARAMETER A FILE-NEVHEZ
1330 : 0AEF 4C C0 FF JMP $FFC0 ;OPEN
1340 :
1350 : *****;A BLOCK-READ/WRITE UTASITAS KULDESE
1360 : 0AF2 BD 63 0B STA $0B63 ;HELY "1" VAGY "2"-NEK
1370 : 0AF5 AD 04 09 LDA $0904 ;LEMEZSZAM
1380 : 0AF9 20 89 0B JSR $0B89 ;A KOD ATVALTASA ASCII-RA
1390 : 0AFB BD 67 0B STA $0B67 ;ES TAROLASA
1400 : 0AFE AD 03 09 LDA $0903 ;A SAVSZAM
1410 : 0B01 20 89 0B JSR $0B89 ;ATVALTASA ASCII-RA,
1420 : 0B04 BE 69 0B STX $0B69 ;ES TAROLASA
1430 : 0B07 BD 6A 0B STA $0B6A
1440 : 0B0A AD 02 09 LDA $0902 ;A SZEKTORSZAM
1450 : 0B0D 20 89 0B JSR $0B89 ;ATVALTASA ASCII-RA,
1460 : 0B10 BE 6C 0B STX $0B6C ;ES TAROLASA
1470 : 0B13 BD 6D 0B STA $0B6D
1480 : 0B16 A9 02 LDA #$02 ;2. KISERLET
1490 : 0B18 BD 01 09 STA $0901
1500 : 0B1B 20 F4 0B JSR $0BF4 ;CHKOUT, KIIRASA 15-OS CSATORNARA
1510 : 0B1E A0 0D LDY #$0D ;13 KARAKTER
1520 : 0B20 FD 62 0B LDA $0B62,X ;"U1: 2 0 TT SS" ILLETVE "U2: 2 0 TT SS" A
1530 : 0B23 20 D2 FF JSR $FFD2 ;LEMEZHEZ
1540 : 0B26 E8 INX
1550 : 0B27 88 DEY
1560 : 0B28 D0 F6 BNE $0B20
1570 : 0B2A 20 CC FF JSR $FFCC ;CLRCH
1580 : 0B2D 20 CE 0B JSR $0BCE ;A HIBACSATORNA OLVASASA
1590 : 0B30 F0 0B BEQ $0B3D ;OK
1600 : 0B32 CE 01 09 DEC $0901 ;KESZ A MASODIK KISERLET
1610 : 0B35 F0 0E BEQ $0B45 ;IGEN
1620 : 0B37 20 97 0B JSR $0B97 ;LEMEZ INICIALIZALASA
1630 : 0B3A 4C 1B 0B JMP $0B1B ;ISMETELT KISERLET
1640 : 0B3D A9 00 LDA #$00 ;'OK' KAPCSOLO
1650 : 0B3F BD 01 09 STA $0901 ;TAROLASA
1660 : 0B42 4C CC FF JMP $FFCC ;CLRCH, KIIRAS ISMET ALAPERTELMEZES SZERINT
1670 : 0B45 A9 FF LDA #$FF ;HIBA-KAPCSOLO
1680 : 0B47 D0 F6 BNE $0B3F ;TOVABB MINT FENT
1690 :
1700 : *****;A 6-OS UTASITAS, LEMEZ FORMALASA
1710 : 0B49 20 F4 0B JSR $0BF4 ;OPEN 15,8,15 ES CHKOUT
1720 : 0B4C A0 10 LDY #$10 ;16 KARAKTER
1730 : 0B4E BD 70 0B LDA $0B70,X ;"N0:CP/M DISK,65"
1740 : 0B51 20 D2 FF JSR $FFD2 ;A LEMEZRE FELIRNI !
1750 : 0B54 E8 INX
1760 : 0B55 88 DEY
1770 : 0B56 D0 F6 BNE $0B4E
1780 : 0B58 20 CC FF JSR $FFCC ;CLRCH

```

```

1790 : 0B5B 20 CE 0B JSR $0BCE ;A KIMENETI CSATORNA OLVASASA
1800 : 0B5E 00 E5 PNE $0B45 ;A HIBAKAPCSOLO BEALLITASA
1810 : 0B60 F0 DB BEQ $0B3D ;AZ OK KAPCSOLO BEALLITASA
1820 :
1830 : *****;SZOVEG LEMEZEKEZELESHEZ
1840 : 0B62 55 31 3A 32 20 30 20 54 54 20 53 53 0D "U1:2 0 TT SS"
1850 : 0B6F 23 ;"#" KOZVETLEN ELERESU FILE
1860 : 0B70 4E 30 3A 43 50 2F 4D 20 44 49 53 4B 2C 36 35 0D "N0:CP/M DISK,65"
1870 : 0B80 42 2D 50 20 32 20 30 0D "B-P 2 0"
1880 : 0B88 49 ;"I"
1890 :
1900 : *****;AZ AKKUBAN LEVO HEXASZAM ATV. ASCII-RA
1910 : 0B89 D8 CLD ;"0"
1920 : 0B8A A2 30 LDX ##30 ;"0"
1925 : 0B8C 38 0A SEC
1930 : 0B9D E9 0A SBC ##0A ;10 KIVONASA
1940 : 0B9F 90 03 BCC $0B94
1950 : 0B91 E8 INX ;AZ X TARTALMAZZA A TIZEDESEKET
1960 : 0B92 B0 F9 BCS $0B8D
1970 : 0B94 69 3A ADC ##3A ;AZ AKKU TARTALMAZZA A SZAMJEGYEKET
1980 : 0B96 60 RTS
1990 :
2000 : *****;INICIALIZALAS ES A KOZVETLEN ELERESU FILE
2005 : ;NYITASA
2010 : 0B97 A9 0F LDA ##0F ;A LOGIKAI FILE-SZAM
2020 : 0B99 20 C3 FF JSR $FFC3 ;CLOSE 15
2030 : 0B9C A9 0F LDA ##0F ;A LOGIKAI FILE-SZAM 15
2040 : 0B9E A2 08 LDX ##08 ;A KESZULEKSZAM 8
2050 : 0BA0 A0 0F LDY ##0F ;15-OS MASODLAGOS CIM
2060 : 0BA2 20 BA FF JSR $FFBA ;A FILE-PARAMETER BEALLITASA
2070 : 0BA5 A9 01 LDA ##01 ;A FILE NEV HOSSZA 1
2080 : 0BA7 A2 88 LDX ##88
2090 : 0BA9 A0 0B LDY ##0B ;MUTATO AZ "I"-RE
2100 : 0BAB 20 BD FF JSR $FFBD ;A FILE-NEV PARAMETEREI
2110 : 0BAE 20 C0 FF JSR $FFC0 ;OPEN15,8,15,"I"
2120 : 0BB1 A9 02 LDA ##02 ;A LOGIKAI FILE-SZAM 2
2130 : 0BB3 20 C3 FF JSR $FFC3 ;CLOSE 2
2140 : 0BB6 A9 LDA ##02 ;A LOGIKAI FILE-SZAM 2
2150 : 0BB8 A2 08 LDX ##08 ;A KESZULEKSZAM 8
2160 : 0BBA A0 02 LDY ##02 ;A 8-AS MASODLAGOS CIM
2170 : 0BBC 20 BA FF JSR $FFBA ;A FILE-PARAMETER BEALLITASA
2180 : 0BBF A9 01 LDA ##01 ;A FILE-NEV HOSSZA 1
2190 : 0BC1 A2 6F LDX ##6F
2200 : 0BC3 A0 0B LDY ##0B ;MUTATO A "*" -RA
2210 : 0BC5 20 BD FF JSR $FFBD ;A FILE-NEV PARAMETEREINEK A BEALLITASA
2220 : 0BC8 4C C0 FF JMP $FFC0 ;OPEN 2,8,2,"#"
2230 : 0BCB 20 97 0B JSR $0B97 ;A MEGHAJTO INICIALIZALASA
2240 :
2250 : *****;A HIBACSATORNA LEOLVASASA
2260 : 0BCE A2 0F LDX ##0F ;A LOGIKAI FILE-SZAM 15
2270 : 0BD0 20 C6 FF JSR $FFC6 ;CHKIN
2280 : 0BD3 B0 F6 BCS $0BCB ;HIBA, MAJD INICIALIZALAS
2290 : 0BD5 20 CF FF JSR $FFCF ;EGY KARAKTER A HIBACSATORNAROL
2300 : 0BD8 C9 30 CMP ##30 ;EGYENLO NULLAVAL
2310 : 0BDA 60 RTS
2320 : 0BDB 20 B1 0B JSR $0BB1 ;A KOZVETLEN ELERESU CSATORNA ISMETELT
2325 : ;NYITASA
2330 :
2340 : *****;BEOLVASASHOZ A KOZVETLEN ELERESU CSATORNA
2345 : ;BEALLITASA
2350 : 0BDE A2 02 LDX ##02 ;A LOGIKAI FILE-SZAM 2
2360 : 0BE0 20 C6 FF JSR $FFC6 ;CHKIN
2370 : 0BE3 B0 F6 BCS $0BDB ;HIBA, NYITAS UJRA
2380 : 0BE5 60 RTS
2390 : 0BE6 20 B1 0B JSR $0BB1 ;A KOZVETLEN ELERESU CSATORNA ISMETELT
2395 : ;MEGNYITASA
2400 :

```

```

2410 : *****;A KOZVETLEN ELERESU CSATORNA BEALLITASA
2415 : ;IRASRA
2420 : 0BE9 A2 02 LDX #02 ;A LOGIKAI FILE-SZAM 2
2430 : 0BEB 20 C9 FF JSR $FFC9 ;CHKOUT
2440 : 0BEE B0 F6 BCS $0BE6 ;HIBA, A CSATORNA ISMETELT MEGNYITASA
2450 : 0BF0 60 RTS
2460 : 0BF1 20 97 0B JSR $0B97 ;A MEGHAJTO INICIALIZALASA
2470 :
2480 : *****;A PARANCSCSATORNA BEALLITASA KIIRASRA
2490 : 0BF4 A2 0F LDX #0F ;A LOGIKAI FILE-SZAM 15
2500 : 0BF6 20 C9 FF JSR $FFC9 ;CHKOUT
2510 : 0BF9 B0 F6 BCS $0BF1 ;HIBA, INICIALIZALAS
2520 : 0BFB A2 00 LDX #00 ;A MUTATO BEALLITASA 0-RA
2530 : 0BFD 60 RTS

```

0 REM **** P56. ****

```

1 :
2 :
100 : *****;A FUNKCIOBILLENTYUK SZOVEGENEK 200-AS CIME
110 : 0C00 10 FC 20 FC 30 FC 40 FC
120 : 0C00 50 FC 60 FC 70 FC 80 FC
125 :
130 : *****;A FUNKCIOBILLENTYUK LEFOGLALASA
140 : 0C10 44 49 52 00 00 00 00 00 00 00 00 00 00 00 00 DIR
150 : 0C20 44 49 52 20 42 3A 0D 00 00 00 00 00 00 00 00 DIR B1
160 : 0C30 53 54 41 54 20 2A 2E 2A 0D 00 00 00 00 00 00 STAT *.*
170 : 0C40 53 54 41 54 20 42 3A 2A 2E 2A 0D 00 00 00 00 STAT B1.*
180 : 0C50 43 4F 50 00 00 00 00 00 00 00 00 00 00 00 00 COPY
190 : 0C60 43 4F 4E 46 48 47 0D 00 00 00 00 00 00 00 00 CONFIG
200 : 0C70 44 44 54 0D 00 00 00 00 00 00 00 00 00 00 00 DDT
210 : 0C80 44 44 54 00 00 00 00 00 00 00 00 00 00 00 00 DDT
220 :
230 : *****;BILLENTYUZET-KODOK TABLAZATA
240 :
250 :
260 : 0D00 08 08 08 10 DEL BS CAN
270 : 0D04 0D 0D 0D 0D CR CR
280 : 0D08 1C 1C 1D 1D KURZOR JOBBRA F8 08
290 : 0D0C 86 86 87 87 F7, F8 F7 F8
300 : 0D10 80 80 81 81 F1, F2 F1 F2
310 : 0D14 82 82 83 83 F3, F4 F3 F4
320 : 0D18 84 84 85 85 F5, F6 F5 F6
330 : 0D1C 1E 1E 1F 1F KURZOR LE RS US
340 : 0D20 33 33 23 33 3 # 3 3 # 3
350 : 0D24 77 57 57 17 W W W W W W
360 : 0D28 61 41 41 01 A A A A A A
370 : 0D2C 34 34 24 34 4 * 4 4 * 4
380 : 0D30 7A 5A 5A 1A Z Z Z Z Z Z
390 : 0D34 73 53 53 13 S S S S S S
400 : 0D38 65 45 45 05 E E E E E E
410 : 0D3C 00 00 00 00 SHIFT NULLA
420 : 0D40 35 35 25 35 5 % 5 5 % 5
430 : 0D44 72 52 52 12 R R R R R R
440 : 0D48 64 44 44 04 D D D D D D
450 : 0D4C 36 36 26 36 6 & 6 6 & 6
460 : 0D54 86 46 46 06 F F F F F F
470 : 0D58 74 54 54 14 T T T T T T
480 : 0D5C 76 56 56 16 X X X X X X
490 : 0D60 37 37 27 37 7 ' 7 7 ' 7
500 : 0D64 78 58 58 18 Y Y Y Y Y Y
510 : 0D68 67 47 47 07 G G G G G G
520 : 0D6C 38 38 28 78 B < B B < B B
530 : 0D70 62 42 42 02 H H H H H H
540 : 0D74 68 48 48 08 U U U U U U
550 : 0D78 75 55 55 15 V V V V V V
560 : 0D7C 76 56 56 16 9 > 9 9 > 9
570 : 0D80 39 39 29 7D

```

| | | | | | | | | | | | |
|-----|---|------|----|----|----|----|------------|---------------|-----|-----|-------|
| 580 | : | 0084 | 68 | 48 | 48 | 08 | I | I | I | I | ↑I |
| 590 | : | 0088 | 5A | 4A | 4A | 0A | J | J | J | J | ↑J |
| 600 | : | 009C | 38 | 38 | 38 | 00 | 0 | 0 | 0 | 0 | NULLA |
| 610 | : | 0080 | 6D | 4D | 4D | 0D | M | M | M | M | ↑M |
| 620 | : | 0094 | 5B | 4B | 4B | 0B | K | K | K | K | ↑K |
| 630 | : | 0098 | 5F | 4F | 4F | 0F | O | O | O | O | ↑O |
| 640 | : | 009C | 5E | 4E | 4E | 0E | N | N | N | N | ↑N |
| 650 | : | 0DA0 | 2B | 2B | 2B | 2B | + | + | + | + | |
| 660 | : | 0DA4 | 7B | 5B | 5B | 1B | P | P | P | P | ↑P |
| 670 | : | 0DAB | 6C | 4C | 4C | 0C | L | L | L | L | ↑L |
| 680 | : | 0DAC | 2D | 2D | 2D | 2D | - | - | - | - | |
| 690 | : | 0DB0 | 2E | 2E | 3E | 7D | . | . | . | . | > |
| 700 | : | 0DB4 | 3A | 3A | 5B | 7B | [| [| [| [| |
| 710 | : | 0DB8 | 4B | 4B | 4B | 6B | @ | @ | @ | @ | |
| 720 | : | 0DBC | 2C | 2C | 3C | 7B | , | , | , | , | < |
| 730 | : | 0DC0 | 5C | 5C | 5C | 7C | / | / | / | / | |
| 740 | : | 0DC4 | 2A | 2A | 2A | 2A | * | * | * | * | |
| 750 | : | 0DC8 | 3B | 3B | 5D | 7D |]] |]] |]] |]] | |
| 760 | : | 0DCC | 1B | 1B | 1B | 7F | HOME | ESC | DEL | | |
| 770 | : | 0DD0 | 00 | 00 | 00 | 00 | SHIFT | NULLA | | | |
| 780 | : | 0DD4 | 3D | 3D | 3D | 3D | = | = | = | = | |
| 790 | : | 0DD8 | 5E | 5E | 5E | 7E | ↑ | ↑ | ↑ | ↑ | |
| 800 | : | 0DDC | 2F | 2F | 3F | 5C | / " ? * | / | / | / | |
| 810 | : | 0DE0 | 31 | 31 | 21 | 31 | ! ! | ! ! | ! ! | ! ! | |
| 820 | : | 0DE4 | 3F | 3F | 3F | 5F | BALRA NYIL | - | - | - | |
| 830 | : | 0DE8 | 00 | 00 | 00 | 00 | CTRL | NULLA | | | |
| 840 | : | 0DEC | 32 | 32 | 22 | 32 | 2 * | 2 | 2 | 2 | |
| 850 | : | 0DF0 | 20 | 20 | 20 | 20 | SZOKOZ | URESZKARAKTER | | | |
| 860 | : | 0DF4 | 00 | 00 | 00 | 00 | COMMODORE | NULLA | | | |
| 870 | : | 0DF8 | 71 | 51 | 51 | 11 | Q | Q | Q | Q | ↑Q |
| 880 | : | 0DFC | 03 | 03 | 03 | 03 | STOP | ↑C | ↑C | ↑C | ↑C |

6.7.3 A Z80 BOOT-rutin

```

0 REM **** P57. ****
1 :
2 :
100 : ; Z80-AS BOOT-RUTIN A C64-EN
110 : ;
120 : ; COPYRIGHT (C) 1982
130 : ; COMMODORE INTERNATIONAL
140 : ;
150 : ; EZT A RUTINT A C64-ES CF/M LEMEZ 1-ES SAV 5-OS
160 : ; SZEKTORABOL A BIOS65 EGY RUTINJA TOLTI BE.
170 : ;
180 : ; A BETOLTESI CIM 0000H (Z80)
190 : ; AMIKOR A Z80-AS BEKAPCSOL EZ A RUTIN BETOLTI A
200 : ; BIOS-T, A CCP-T A RAM-BA, MAJD RAUGRIK.
210 : ;
220 : 3400 = CCP EQU 3400H ;A Z80-S CF/M KEZDOCSIME
230 : ; CCP EQU 0000H ;BOOT0.HEX
240 : ; CCP EQU 0100H ;BOOT1.HEX
250 : 001C = NSECTS EQU 1CH ;A BETOLTENDO SZEKTOROK SZAMA
260 : F903 = TRACK EQU 0F903H ;A SAVSZAM REGISZTERE
270 : F902 = SECTOR EQU 0F902H ;A SZEKTORSZAM REGISZTERE
280 : F904 = DISKNO EQU 0F904H ;A LEMEZSZAM REGISZTERE
290 : FCFE = IOTYPE EQU 0FCFEH ;10 KONFIGURACIO A BIOS65-BEN
300 : 4A33 = KYBDMD EQU CCP+1633H ;NAGYBETUS MOD KAPCSOLO
310 : 0000 = VICRD EQU 0 ;A SZEKTOR OLVASASANAK UTASITASKODJAA
320 : F900 = CMD EQU 0F900H ;UTASITAS REGISZTER
330 : 0001 = OFF EQU 01H ;A Z80-AS KIKAPCSOLASA
340 : CE00 = MODESW EQU 0CE00H ;A Z80-AS KIKAPCSOLASANAK CIME
350 : F901 = DATA EQU 0F901H ;AZ ADATATVITELI REGISZTER
360 : F800 = BUFFER EQU 0F800H ;LEMEZPUFFER
370 : 4A00 = BOOT EQU CCP+1600H ;A BIOS BOOT CIME

```

```

375 :
380 : 0000
390 :
400 : 0000 00
410 : 0000 110034
420 : 0004 3E00
430 : 0006 3204F9
440 : 0009 2601
450 : 000B 2E06
460 : 000D 7C
470 : 000E 3203F9
480 : 0011 7D
490 : 0012 3202F9
500 : 0015 3E00
510 : 0017 3000F9
520 : 001A 3E01
530 : 001C 3200CE
540 : 001F 00
550 : 0020 3A01F9
560 : 0023 B7
570 : 0024 C20D00
575 :
580 :
590 :
600 : 0027 3E2A
610 : 0029 3201F9
620 : 002C 3E03
630 : 002E 3200F9
640 : 0031 3E01
650 : 0033 3200CE
660 : 0036 00
670 :
680 :
690 :
700 : 0037 0100F0
710 : 003A 0A
720 : 003B 12
730 : 003C 0C
740 : 003D 1C
750 : 003E C23A00
760 :
770 :
780 :
790 : 0041 14
800 : 0042 2C
810 : 0043 7D
820 :
830 :
840 :
850 : 0044 FE11
860 : 0046 DA4C00
870 : 0049 24
880 : 004A 2E00
890 : 004C 7C
900 :
910 :
920 :
930 : 004D FE03
935 : 004F C20D00
940 :
945 : 0052 3AFFFC
950 : 0055 E620
955 : 0057 C25F00
960 : 005A 3E01
965 : 005C 32334A
970 :
975 :
980 : 005F C3004A

;
ORG 0000H ;A Z00 RESET CIM

NOP ;A HARDVERHEZ KELL !
LXI D,CCP ;AZ ELSO BETOLTESI CIM
MVI A,0
STA DISKNO ;A BETOLTES AZ A EBYSEBROL
MVI H,1 ;START AZ 1-ES SAV 6-OS SZEKTORTOL
MVI L,6
LOAD1 MOV A,H
STA TRACK
MOV A,L
STA SECTOR
MVI A,VICRD ;A SZEKTOR-OLVASAS UTASITAS
STA CHD
MVI A,OFF
STA MODESW ;A Z00 BEKAPCSOLASA
NOP
LDA DATA ;AZ ATVITEL RENDBEN
ORA A
JNZ LOAD1 ;MEGEGYSZER, HA NEM

;
;
;
EBY "*" KIIRASA SZEKTORONKENT

MVI A,'*'
STA DATA
MVI A,3 ;KIIRAS A KEPERNYORE
STA CHD
MVI A,OFF
STA MODESW ;ATVITEL A 6510-ESHEZ
NOP

;
;
;
AZ OLVASOTT SZEKTOR ATVITELE A TARBA

LXI B,BUFFER ;A BC LEMEZPUFFER MUTATOJA
LOAD2: LDAX B ;EGY BYTE A PUFFERBOL
STAX D ;A TARBA
INR C
INR E ;A MUTATO NOVELESE
JNZ LOAD2

;
;
;
A SAV/SZEKTOR-SZAM NOVELESE

INR D
INR L ;A SZEKTORSZAM NOVELESE ES BETOLTESE
MOV A,L ;AZ AKKUBA

;
;
;
A SAV VEGENEK VIZSGALATA

CPI 17 ;ELERTUK A 17-ES SAVOT
JC LOAD3 ;NEM
INR H ;A SAVSZAM NOVELESE
MVI L,0 ;A SZEKTORSZAM NULLA
LOAD3 MOV A,H ;A SAVSZAM AZ AKKUBA

;
;
;
KESZ,HA A'H'A 3-AS SAVRA MUTAT

CPI 3
JNZ LOAD1 ;A KOVETKEZO SZEKTOR OLVASASA

;
;
;
LDA IOTYPE ;NAGY/KISBETU MOD
ANI 20H
JNZ LOAD4
MVI A,1
STA KYBMD ;A KAPCSOLO BEALLITASA A BIOS-BAN

;
LOAD4: ;UGRAS A BIOS BOOT RESZERE
JMP BOOT

```


6.7.4 A BIOS a Commodore 64-esen CP/M 2.2-re

```

0 REM **** P58. ****
1 :
2 :
100 :      %      COPYRIGHT (C) 1982
110 :      %      COMMODORE INTERNATIONAL
120 :      %
130 :      %      A VALTOZAT SAJATOSSAGAI:
140 :      %
150 :      %      1. 52 K RAM IGENYU BIOS RENDSZER, A BIOS 65 ALTAL
160 :      %      LETREHOZOTT I/O RUTINNAL.
170 :      %
180 :      %      2. LEMEZTABLAZATOKAT ES A VEKTOROKAT A 2-ES
190 :      %      EGYSEG TAROLJA.
200 :      %
210 :      %      3. I/O BYTE NINCS BEEPITVE
220 :      %
230 :      %      4. LYUKASZTO ES OLVASO RUTINOK MEG NINCSENEK.
240 :      %
250 :      %      5. A KLAVIATURA ES A FUNKCIOBILLENTYUK
260 :      %      HOZZARENDELESET A BIOS 65 TARTALMAZZA.
270 :      %
280 :      %      6. A COMMODORE 64-ES 20K-48K-IG TERJEDO CP/M
290 :      %      RENDSZERT TAMOGAT.
300 :      %
310 :      %      7. VIRTUALIS LEMEZEZEGYSEG (B) A VC 1541 MELLETT
320 :      %
330 :      %      8. VALODI LEMEZEZEGYSEG (B) AZ IEC-BUS MELLETT
340 :      %
350 :      %
360 : 0000 = BASE EQU 0000H ;A CIMEZHETO RAM KEZDETE
370 :      %
380 : 002C = MSIZE EQU 44 ;A CP/M TARIKENYE KBYTE-BAN
390 :      %
400 :      %      "BIAS" A TAR 3400H-TOL SZAMITOTT RELATIV (OFFSET)
410 :      %      CIME (A SZOVEGBEN 'B'-VEL JELOLVE)
420 :      %
421 : 6000 = BIAS EQU (MSIZE-20)*1024
422 :      %
423 :      %      UTMUTATO: A MOVCPM ELOALLITASAHOZ A
424 :      %      KOVETKEZO UTASITASOKRA VAN SZUKSEG :
425 :      %
430 :      % CCP EQU 0000H ;BIOS0 HEX
440 :      % CCP EQU 0100H ;BIOS1 HEX
450 :      %
460 : 7400 = CCP EQU 3400H+BIAS ;A CCP BAZISCIME
470 : 7C06 = BDOS EQU CCP+806H ;A BDOS BAZISCIME
480 : AA00 = BIOS EQU CCP+1600H ;A BIOS BAZISCIME
490 : 0004 = CDISK EQU BASE+0004H ;AZ AKTUALIS EGYSEG SZAM
495 :      % ;0...15 EGYENLO A...F
500 : 0003 = IOBYTE EQU BASE+0003H ;INTEL I/O BYTE
510 : 0000 = TRANS EQU 0000H ;HA 0, NINCS FORDITAS
515 : 0005 = ENTRY EQU 0005H ;BDOS UBRASVEKTOR
520 :      %
530 :      %      A KOVETKEZO UTASITASOK MEGHATAROZZAK A TELJES
540 :      %      TARTERULETET A 6510-ES I/O RUTINOK ADATFORGALMAHOZ
550 :      %
570 : F800 = HSTBUF EQU 0F800H ;256 BYTE LEMEZIPUFFER
580 : F900 = CMD EQU 0F900H ;UTASITASREGISZTER
590 : F901 = DATA EQU 0F901H ;ADATREGISZTER
600 : F902 = SECTOR EQU 0F902H ;SZEKTORREGISZTER
610 : F903 = TRACK EQU 0F903H ;SAVREGISZTER
620 : F904 = DISKNO EQU 0F904H ;EGYSEG SZAM REGISZTER
630 : F905 = KYCHAR EQU 0F905H ;BILLENTYUKOD REGISZTER
640 : FCFF = IOTYPE EQU 0FCFFH ;I/O KONFIGURACIO-BYTE

```

```

650 : ;
660 : ; A Z80-AS A 'MODESW' TARCIMRE 'OFF'-T IRVA
670 : ; KIKAPCSOLJA ONMAGAT
680 : ;
690 : 0001 = OFF EQU 1
700 : CE00 = MODESW EQU 0CE00H
710 : ;
720 : ; A 6510-ES I/O RUTINJA
730 : ;
740 : 0000 = VICRD EQU 0 ;A KIVALASZTOTT SZEKTOR OLVASASA
750 : 0001 = VICWR EQU 1 ;A KIVALASZTOTT SZEKTOR IRASA
760 : 0002 = VICIN EQU 2 ;A KLAVIATURA LEKERDEZESE
770 : 0003 = VICOUT EQU 3 ;KIIRAS KEPERNYORE
780 : 0004 = VICPST EQU 4 ;A NYOMTATO STATUSZANAK BETOLTESE
790 : 0005 = VICPRT EQU 5 ;KIIRAS NYOMTATORA
800 : 0006 = VICFMT EQU 6 ;LEMEZFORMALAS
810 : 0007 = AUX1 EQU 7 ;UGRAS A 6510-ES $E00 CIMRE
820 : 0008 = AUX2 EQU 8 ;UGRAS A 6510-ES $F00 CIMRE
830 : 0009 = INDIR EQU 9 ;INDIREKT UGRAS ($906) SZERINT
840 : ;
850 : ;
860 : 8600 = AA00 ORG BIOS ;PROGRAM STARTOK
870 : 0016 NSECTOR EQU ($-CCP)/256 ;A MELEGSTART SZEKTORSZAMLAOLJA
880 : ;
890 : ; UGRASVEKTOROK AZ EGYES RUTINOKRA
900 : AA00 C36CAA JMP BOOT ;HIDEQ START
910 : AA03 C31DAB WBOOT: JMP WBOOT ;MELEG START
920 : AA06 C39EAB JMP CONST ;KONZOLSTATUSZ
930 : AA09 C30AAC JMP CONIN ;1 KARAKTER BETOLTESE A KONZOLROL
940 : AA0C C383AC JMP CONOUT ;1 KARAKTER KIIRASA A KONZOLRA
950 : AA0F C3C8AC JMP LIST ;1 KARAKTER A LIST-EGYSEGRE
960 : AA12 C316AD JMP PUNCH ;1 KARAKTER A PUNCH EGYSEGRE
970 : AA15 C319AD JMP READER ;1 KAR. BEOLV. A READER EGYSEGBOL
980 : AA18 C31EAD JMP HOME ;AZ OLVASOFFJ BEALLITASA
985 : ; ;HOME POZICIOBA
990 : AA1B C329AD JMP SELDSK ;LEMEZ VALASZTAS
1000 : AA1E C33DAD JMP SETTRK ;SAVSZAM BEALLITAS
1010 : AA21 C343AD JMP SETSEC ;SZEKTORSZAM BEALLITAS
1020 : AA24 C34BAD JMP SETDMA ;A DMA-CIM BEALLITASA
1030 : AA27 C351AD JMP READ ;OLVASAS A LEMEZROL
1040 : AA2A C365AD JMP WRITE ;IRAS A LEMEZRE
1050 : AA2D C3EBAC JMP LISTST ;A LIST-EGYSEG STATUSZANAK
1055 : ; ;BETOLTESE
1060 : AA30 C34EAD JMP SECTRAM ;SZEKTORSZAM ELTOLASA
1070 : ;
1080 : AA33 00 KYBDMD: DB 00H ;NAGYBETUS UZEMMOD KAPCSOLO
1090 : ;
1100 : ; KET LEMEZEGYSEG TABLAZATA
1110 : ; A 0-AS EGYSEG LEMEZPARAMETERE (HEADER)
1120 : AA34 00000000 DPMASE: DW TRANS,0000H
1130 : AA38 00000000 DW 0000H,0000H
1140 : AA3C 1FAF54AA DW DIRBF,DPBLK
1150 : AA40 DD4F9FAF DW CHK00,ALL00
1160 : ; AZ 1-ES EGYSEG LEMEZPARAMETERE (HEADER)
1170 : AA44 00000000 DW TRANS,0000H
1180 : AA48 00000000 DW 0000H,0000H
1190 : AA4C 1FAF54AA DW DIRBF,DPBLK
1200 : AA50 ED4FBEAF DW CHK01,ALL01
1210 : ;
1220 : ;
1230 : ; DPBLK: ;AZ ALTALANOS LEMEZRE VONATKOZO PARAMETEREK
1240 : AA54 2200 DW 34 ;SAVONKENTI SZEKTORSZAM
1250 : AA56 03 DB 3 ;BLOKKELTOLASI TENYEZO
1260 : AA57 07 DB 7 ;BLOKK MASZK
1270 : AA58 00 DB 0 ;NULLA MASZK
1280 : AA59 8700 DW 135 ;LEMEZKAPACITAS

```

```

1290 : AA5B 3F00      DW 63      ;A FILE-BEJEGYZESEK MAX. SZAMA
1300 : AA5D C0        DB 192     ;FOGLALTSAG (0)
1310 : AA5E 00        DB 0       ;FOGLALTSAG (1)
1320 : AA5F 1000      DW 16      ;FELULVIZSGALT BEJEGYZESEK
1330 : AA61 0200      DW 2       ;SAV OFFSET
1340 : ;
1350 : ;
1360 : ; A TABLAZAT VEGE
1370 : ;
1380 : ; A KOVETKEZO TARTERULETEKET A RENDSZER
1390 : ; BETOLTESEKOR INICIALIZALJA
1400 : AA63 40        LASTKY: DB 40H      ;AZ UTOLJARA BEUTOTT BILL.VEKTORA
1410 : AA64 00        TOGGLE: DB 00H      ;ATKAPCSOLAS NAGYBETUS UZEMMODRA
1420 : AA65 00        CSTAT: DB 00H      ;A KESZ KARAKTER KAPCSOLOJA
1430 : AA66 0000      MSGPTR: DW 0000H   ;MUTATO A FUNKCIO BILL. SZOV.-ERE
1440 : AA68 00FD      TBLPTR: DW 0FD00H  ;BILLENTYUZET-DEKODOLO TABLAZAT
1450 : AA6A 00FC      MSGTBL: DW 0FC00H  ;A FUNKCIO BILLENTYUK VEKTORA
1460 : ;
1470 : ; EGYEB KLAVIATURA-MEGHATAROZASOK
1480 : ;
1490 : F28D =         SHFTST EQU 0F28DH   ;CONTROL,COMMODORE ES SHIFT BILL.
1500 : F0CC =         FLASH EQU 0F0CCH   ;A KURZOR BEKAPCSOLASA
1510 : F0CF =         CURSOR EQU 0F0CFH   ;KURZORKARAKTER
1520 : ;
1530 : ; A MUVELETEKET ELVESZO RUTINOK
1540 : BOOT:
1550 : AA6C 3E20        MVI A,20H      ;AZ ASCII URES KARAKTER,
1560 : AA6E 32CFF0      STA CURSOR      ;MINT KURZORKARAKTER
1570 : AA71 AF          XRA A          ;AZ AKKU TORLESE
1580 : AA72 320300     STA I0BYTE     ;AZ I/O BYTE TORLESE
1590 : AA75 320400     STA CDISK       ;A 0-AS LEMEZEZEGYSEG KIVALASZTASA
1600 : AA78 321EAF      STA CURDSK    ;MUTATO A VIRTUALIS LEMEZRE
1610 : AA7B 3210AF      STA HSTACT     ;A PUFFER INAKTIV
1620 : AA7E 3212AF      STA UNACNT    ;A 'NEM FOGLALT' SZAMLALO TORLESE
1630 : AA81 3EC3       MVI A,0C3H     ;A C3: A JUMP OPCODE KODJA
1640 : AA83 320000     STA 0+BASE      ;UGRAS A WBOOT-RA
1650 : AA86 2103AA     LXI H,WBOOTE    ;A WBOOT UGRASI CIMENEK
1660 : AA89 220100     SHLD 1+BASE    ;BEALLITASA
1670 : ;
1680 : AA8C 320500     STA 5+BASE      ;UGRAS A BDOS-RA
1690 : AA8F 21069E     LXI H,BDOS      ;A BDOS UGRASI CIMENEK
1700 : AA92 220600     SHLD 6+BASE    ;BEALLITASA
1710 : ;
1720 : AA95 010800     LXI B,80H+BASE   ;ALAPERTELMEZES SZERINTI DMA CIM
1730 : AA98 CD4BAD      CALL SETDMA
1740 : ;
1750 : AA9B 11A6AA      LXI D,SIGNON    ;A 'DE' A BEJELENTEKEZESI UZENETRE
1755 : ;
1760 : AA9E 0E09        NVI C,9        ;A FUZERKIIRO MUVELET
1770 : AAA0 CD0500     CALL ENTRY      ;A BDOS-BAN
1780 : AAA3 C38DAF     JMP 60CPM1     ;KESZ A CCP SZAMARA
1790 : ;
1800 : AAA6 0C0A        SIGNON: DB 0CH,0AH ;KEPERNYOTORLES
1810 : AAA8 2020202043  DB 'COMMODORE-64 20K CP/M 2.2 VALTOZAT'
1820 : AAC0 0D0A00      DB 0DH,0AH,0AH
1830 : AACF 2020406F70  DB 'COPYRIGHT (C) 1979 DIGITAL RESEARCH',0DH,0AH
1840 : AAF7 2020702020  DB 'COPYRIGHT (C) 1982 COMMODORE',0DH,0AH
1850 : AB1B 0A24        DB 0AH,'$'    ;A FUZER VEGET JELZO KARAKTER
1860 : ;
1865 : ;
1870 : ;
1880 : ; WBOOT:
1890 : AB1D 318000      LXI SP,80H+BASE ;A PUFFEREN BELULI TARTERULETET
1900 : ; VEREMKENT HASZNALJUK
1910 : AB20 0E0.        MVI C,0        ;A 0. EGYSEG KIVALASZTASA
1920 : AB22 CD29AD      CALL SELDSK

```

```

1930 : AB25 AF          XRA A          ;AZ "A" EGYSEG
1935 : AB26 3204F9     STA DISKNO     ;KIVALASZTASA
1940 : AB29 CDA5AE     CALL CHGDSK    ;HA EZ MEG NEM TORTENT MEG
1950 : AB2C CD1EAD     CALL HOME     ;OLVASOFEJ BEALLITASA A 0. SAVRA
1960 : AB2F 3E0D       MVI A,ODH     ;KOCSI VISSZA
1970 : AB31 CDC0AC     CALL COUTS    ;KIIRASA
1980 :
;
1990 : AB34 110094     LXI D,CCP     ;A BETOLTES INDUL
2000 : AB37 0616       MVI B,NSECTS
2010 : AB39 2601       MVI H,1      ;SAVSZAM
2020 : AB3B 2E06       MVI L,6      ;SZEKTORSZAM
2030 : AB3D 7C         LOAD1: MOV A,H
2040 : AB3E 3203F9     STA TRACK
2050 : AB41 7D         MOV A,L
2060 : AB42 3202F9     STA SECTOR
2070 : AB45 3E00       MVI A,VICRD ;LEMEZOLVASASI UTASITAS
2080 : AB47 CD94AB     CALL IO6510
2090 :
;
2100 : AB4A 3A01F9     LDA DATA
2110 : AB4D B7         ORA A
2120 : AB4E C23DAB     JNZ LOAD1    ;HA NEM 0, AKKOR HIBA
2130 : AB51 E5         PUSH H
2140 : AB52 C5         PUSH B
2150 : AB53 010001     LXI B,256
2160 : AB56 2100F8     LXI H,HSTBUF ;LEMEZPUFFER
2170 : AB59 ED         DB 0EDH     ;A Z80-AS LDIR UTASITASA
2180 : AB5A B0         DB 0B0H
2190 : AB5B 0E2A       MVI C,'*'   ;MINDEN SZEKTOR UTAN EGY '*' KIIRASA
2200 : AB5D CDB3AC     CALL CONOUT
2210 : AB60 C1         POP B
2220 : AB61 E1         POP H
2230 : AB62 05         DCR B       ;A SZEKTORSZAM CSOKKENTESE
2240 : AB63 CA73AB     JZ GOCPM
2250 : AB66 2C         INR L       ;A KOVETKEZO SZEKTOR
2260 : AB67 7D         MOV A,L
2270 : AB68 FE11       CPI 17
2280 : AB6A DA3DAB     JC LOAD1
2290 : AB6D 24         INR H
2300 : AB6E 2E00       MVI L,0
2310 : AB70 C33DAB     JPM LOAD1
2320 :
;
2330 :
;
2340 :
;
2350 : AB73 3EC3       MVI A,0C3H ;C3 A JUMP UTASITAS
2360 : AB75 320000     STA 0+BASE ;UGRAS A WBOOT-RA
2370 : AB78 2103AA     LXI H,WBOOTE ;A WBOOT UGRASI CIME
2380 : AB7B 220100     SHLD 1+BASE ;A CIMMEZO BEALLITASA 0-RA
2390 :
;
2400 : AB7E 320500     STA 5+BASE ;UGRAS A BDOS-RA
2410 : AB81 21069C     LXI H,BDOS  ;A BDOS UGRASI CIME
2420 : AB84 220600     SHLD 6+BASE ;A CIMMEZO BEALLITASA 5-RE
2430 :
;
2440 : AB87 018000     LXI B,80H+BASE ;A DMA KEZDO CIME
2450 :
;
2460 : AB8A CD48AD     CALL SETDMA ;ALAPERTELMEZESBEN 80H
2470 :
;
2480 :
;
2490 : AB8D 3A0400     GOCPM1: LDA CDISK ;AZ EGYSEGSZAM BETOLTESE
2500 : AB90 4F         MOV C,A      ;TOVABBI FELDOLGOZASRA
2510 : AB91 C30094     JMP CCP     ;A CCP-VEL
2520 :
;
2530 :
;
2540 :
;
2550 :
;
2560 : AB94 3200F9     IO6510: STA CMD ;AKKU TARTALMA AZ UTASITASREGISZTERBE
2570 : AB97 3E01       MVI A,OFF

```

```

2580 : AB99 3200CE STA MODESW ;A Z80-AS KIKAPCSOLASA A
2590 : AB9C 00 NOB ;HARDVER MIATT
2600 : AB9D C9 RET
2610 : ;
2620 : ;
2630 : ;
2640 : CONST: ;A KONZOLSTATUSZ FFH, HA A KARAKTER KESZ,
2650 : ;EGYEBKENT 00H
2660 : AB9E 2A66AA LHLD MSGPTR ;FUNKCIOBILLENTYU-UZEMMOD
2670 : ABA1 7C MOV A,H
2680 : ABA2 85 ORA L
2690 : ABA3 3EFF MVI A,0FFH ;"AZ ADAT KESZ" KAPCSOLO
2700 : ABA5 C0 RNZ ;RETURN,HA MSGPTR <> 0
2710 : ;
2720 : ABA6 3A65AA LDA CSTAT ;VOLT-E MAR KARAKTER
2730 : ABA9 A7 ANA A
2740 : ABAA C0 RNZ ;IGEN, HA NEM 0
2750 : ;
2760 : ABAB 3E02 MVI A,VICIN ;EGY KARAKTERT BETOLTO UTASITAS
2770 : ABAB 3E02 CALL I06510
2780 : ;
2790 : AB98 3A8DF2 LDA SHFTST ;A KONTROLLBILLENTYU STATUSZANAK
2795 : ;BETOLTESE
2800 : ABB3 E602 ANI 02A ;A COMMODORE BILLENTYU VIZSGALATA
2810 : ABB5 CAC9AD JZ CONST0 ;UGRAS, NINCS LENYOMVA
2820 : ;
2830 : ABB8 3A64AA LDA TOGGLE ;LE VOLT-E NYOMVA BILLENTYU
2840 : ABB8 A7 ANA A
2850 : ABB8 C2C9AB JNZ CONST0 ;NEM, VARAKOZAS
2860 : ;
2870 : ABBF 3A33AA LDA KYBDMD ;A NAGYBETUS UZEMML. KAPCS. BETOLTESE
2880 : ABC2 EE01 XRI 01H ;AZ UZEMMODBIT KIELEHZESE
2890 : ABC4 3233AA STA KYBDMD
2900 : ABC7 3E01 MVI A,1
2910 : ABC9 3264AA CONST0: STA TOGGLE
2920 : ;
2930 : ABCC 3A05F9 LDA KYCHAR ;EGY KARAKTER BETOLTESE
2940 : ABCF FE3A CPI 3AH ;ERVENYTELEN KONTROLL-KARAKTER
2950 : ABD1 CAE0AB JZ CONST1
2960 : ;
2970 : ABD4 FE3D CPI 3DH ;ERVENYTELEN KONTROLL-KARAKTER
2980 : ABD6 CAE0AB JZ CONST1
2990 : ;
3000 : ABD9 2163AA LXI H,LASTKY;OSSZEHASONLITAS AZ UTOLSO BILL.-VEL
3010 : ABDC BE CMP M ;ADATVIZSGALAT
3020 : ABD0 C2E5AB JNZ CONST2 ;HA KULONBOZO, UJ BILLENTYU
3030 : ;
3040 : ABE0 AF CONST1: XRA A ;"NINCS KESZ ADAT" KAPCSOLO
3050 : ABE1 3265AA STA CSTAT ;TAROLASA
3060 : ABE4 C9 RET
3070 : ;
3080 : ABE5 F5 CONST2: PUSH PSW
3090 : ABE6 01F401 LXI B,500
3100 : ABE9 0B CONST3: DCX B
3110 : ABEA 79 MOV A,C
3120 : ABE8 0B ORA B
3130 : ABEC C2E9AB JNZ CONST3
3140 : ;
3150 : ABEF 3E02 MVI A,VICIN ;MEG EGY KARAKTER BETOLTESE
3160 : ABF1 CD94AB CALL I06510
3170 : ;
3180 : ABF4 F1 POP PSW
3190 : ABF5 2105F9 LXI H,KYCHAR
3200 : ABF8 BE CMP M
3210 : ABF9 C20EAB JNZ CONST1 ;
3220 : ;

```

| | | | | | |
|--------|-------------|---------|--------------------------------------|---|------------------------------------|
| 3230 : | ABFC 3263AA | STA | LASTKY | : | |
| 3240 : | ABFF FE40 | CPI | 40H | : | 40H, HA NINCS LENYOMVA A BILLENTYU |
| 3250 : | AC01 CAE0AB | JZ | CONST1 | : | |
| 3260 : | | | | : | |
| 3270 : | AC04 3EFF | MVI | A,0FFH | : | A "KESZ KARAKTER" KAPCSOLO |
| 3280 : | AC06 3265AA | STA | CSTAT | : | TAROLASA |
| 3290 : | AC09 C9 | RET | | : | |
| 3300 : | | | | : | |
| 3310 : | | CONIN: | EGY KARAKTER BETOLTESE A REGISZTERBE | : | |
| 3320 : | AC0A 3E00 | MVI | A,0 | : | A KURZOR BEKAPCSOLASA |
| 3330 : | AC0C 32CCF0 | STA | FLASH | : | |
| 3340 : | | | | : | |
| 3350 : | AC0F 2A66AA | LHLD | MSGPTR | : | A FUNKCIOBILLENTYU-UZEMMOD |
| 3360 : | AC12 2C | MOV | A,H | : | |
| 3370 : | AC13 B5 | ORA | L | : | |
| 3380 : | AC14 C25FAC | JNZ | CONIN5 | : | |
| 3390 : | | | | : | |
| 3400 : | AC17 CD9EAB | CONIN1: | CALL CONST | : | A KONZOLSTATUSZ VIZSGALATA |
| 3410 : | AC1A B7 | ORA | A | : | |
| 3420 : | AC1B CA17AC | JZ | CONIN1 | : | VARAKOZAS AZ UJ KARAKTERRE |
| 3430 : | | | | : | |
| 3440 : | AC1E AF | XRA | A | : | |
| 3450 : | AC1F 3265AA | STA | CSTAT | : | A STATUSZ TORLESE |
| 3460 : | AC22 3A33AA | CONIN2: | LDA KYBDM | : | SHIFT=0, NAGY=1 |
| 3470 : | AC25 47 | MOV | B,E | : | |
| 3480 : | AC26 3A8DF2 | LDA | SHFTST | : | A SHIFT/CTRL STATUSZ BETOLTESE |
| 3490 : | AC29 E601 | ANI | 01H | : | LE VAN-E NYOMVA A SHIFT |
| 3500 : | AC2B CA30AC | JZ | CONIN3 | : | UGRAS,HA NINCS |
| 3510 : | | | | : | |
| 3520 : | AC2E 0602 | MVI | B,2 | : | SHIFT=2 |
| 3530 : | AC30 3A8DF2 | CONIN3: | LDA SHFTST | : | A STATUSZ BETOLTESE |
| 3540 : | AC33 E604 | ANI | 04H | : | LE VAN-E NYOMVA A KONTROLL |
| 3550 : | AC35 CA3AAC | JZ | CONIN4 | : | UGRAS,HA NINCS |
| 3560 : | | | | : | |
| 3570 : | AC38 0603 | MVI | B,3 | : | KONTROLL=3 |
| 3580 : | AC3A 3A63AA | CONIN4: | LDA LASTKY | : | A BILLENTYUSZAM BETOLTESE |
| 3590 : | AC3D 87 | ADD | A | : | *2 |
| 3600 : | AC3E 87 | ADD | A | : | *4 |
| 3610 : | AC3F 80 | ADD | B | : | A SZAMOK OSSZEADASA |
| 3620 : | AC40 2A68AA | LHLD | TBLPTR | : | A BILLENTYUTABLAZAT CIME |
| 3630 : | AC43 85 | ADD | L | : | MUTATO A TABLAZATRA |
| 3640 : | AC44 6F | MOV | L,A | : | |
| 3650 : | AC45 3E00 | MVI | A,0 | : | |
| 3660 : | AC47 8C | ADC | H | : | |
| 3670 : | AC48 67 | MOV | H,A | : | |
| 3680 : | AC49 7E | MOV | A,M | : | EGY KARAKTER BETOLTESE A TABL.-BOL |
| 3690 : | AC4A FEB0 | COI | 90H | : | HA > 7FH, AKKOR FUNKCIOBILLENTYU |
| 3700 : | AC4C DA71AC | JC | CONIN7 | : | UGRAS,HA ASCII-KARAKTER |
| 3710 : | AC4F 2A6AAA | LHLD | MSGTBL | : | A SZOVEGTABLAZAT KEZDO CIME |
| 3720 : | AC52 E67F | ANI | 7FH | : | 7 BIT TORLESE |
| 3730 : | AC54 87 | ADD | A | : | *2 |
| 3740 : | AC55 85 | ADD | L | : | MUTATO A TABLAZATRA |
| 3750 : | AC56 6F | MOV | L,A | : | |
| 3760 : | AC57 3E00 | MVI | A,0 | : | |
| 3770 : | AC59 8C | ADC | H | : | |
| 3780 : | AC5A 67 | MOV | H,A | : | |
| 3790 : | AC5B 7E | MOV | A,M | : | LO BYTE |
| 3800 : | AC5C 23 | INX | H | : | |
| 3810 : | AC5D 66 | MOV | H,M | : | HI BYTE |
| 3820 : | AC5E 6F | MOV | L,A | : | |
| 3830 : | AC5F 46 | CONIN5: | MOV B,M | : | EGY KARAKTER BETOLTESE |
| 3840 : | AC60 23 | INX | H | : | A KOVETKEZO KARAKTER VIZSGALATA |
| 3850 : | AC61 7E | MOV | A,M | : | |
| 3860 : | AC62 A7 | ANA | A | : | |
| 3870 : | AC63 C269AC | JNZ | CONIN6 | : | HA 0,B-BEN AZ UTolso KARAKTER |
| 3880 : | | | | : | |

```

3890 : AC66 210000
3900 : AC69 2266AA
3910 : AC6C 7B
3920 : AC6D A7
3930 : AC6E CA17AC
3940 :
3950 : AC71 F5
3960 : AC72 3E01
3970 : AC74 32CCF0
3980 : AC77 0E20
3990 : AC79 CD83AC
4000 : AC7C 3E9D
4010 : AC7E CDC0AC
4020 : AC81 F1
4030 : AC82 C9
4040 :
4050 :
4060 : AC83 3AFFFC
4070 : AC86 E610
4080 : AC88 79
4090 : AC89 C2C0AC
4100 :
4110 : AC8C CDF4AC
4120 : AC8F FE0C
4130 : AC91 C299AC
4140 :
4150 : AC94 3E93
4160 : AC96 C3C0AC
4170 :
4180 : AC99 FE08
4190 : AC9B C2A3AC
4200 : AC9E 3E14
4210 :
4220 : ACA0 C3C0AC
4230 : ACA3 FE0A
4240 : ACA5 C2ADAC
4250 :
4260 : ACA8 3E11
4270 : ACAA C3C0AC
4280 :
4290 : ACAD FE0D
4300 : ACAF C2DAAC
4310 :
4320 : ACB2 CDC0AC
4330 : ACB5 3E91
4340 : ACB7 C3C0AC
4350 :
4360 : ACBA FE20
4370 : ACBC D8
4380 : ACBD FE80
4390 : ACBF D8
4400 :
4410 : ACC0 3201F9
4420 : ACC3 3E03
4430 : ACC5 C394AB
4440 :
4450 :
4460 : ACCB 3AFFFC
4470 : ACCB E604
4480 : ACCD 79
4490 : ACCE C2E3AC
4500 :
4510 : ACD1 3AFFFC
4520 : ACD4 E608
4530 : ACD6 79
4540 : ACD7 C2E0AC

CONIN6: LXI H,0000H ;A SZOVEG MOD VEGE
        SHLD MSGPTR ;A SZOVEGMUTATO TAROLASA
        MOV A,B ;A KARAKTER VIZSGALATA
        ANA A ;HA AZ ELOZO JEL 0
        JZ CONIN1 ;HA 0 AKKOR NINCS TOBB KARAKTER
;
CONIN7: PUSH PSW ;A KARAKTER TAROLASA
        MVI A,1
        STA FLASH ;A KURZOR KIKAPCSOLASA
        MVI C, ;URES KARAKTER
        CALL CONOUT ;KIIRAS
        MVI A,9DH ;KURZOR BALRA
        CALL COUT5 ;A SZURO ATUGRASA
        POP PSW ;A KARAKTER VISSZATOLTESE
        RET ;KESZ
;
CONOUT: ;EGY KARAKTER A C REGISZTERBOL A KONZOLRA
        LDA IOTYPE ;KONFIGURACIO-BYTE
        ANI 10H ;A FILTER IGNORALVA
        MOV A,C ;AZ AKKUBAN LEVO KARAKTEREK
        JNZ COUT5 ;KIIRASA VALTOZATLANUL
;
        CALL SWAP ;NAGY/KISBETU FELCSERELESE
        CPI 0CH ;A KEPERNYOTORLES ASCII-KODJA
        JNZ COUT1 ;UGRAS HA NEM
;
        MVI A,93H ;A KEPERNYOTORLES COMMODORE-KODJA
        JMP COUT5
;
COUT1: CPI 08H ;ASCII VISSZALEPES
        JNZ COUT2 ;UGRAS,HA NEM
        MVI A,14H ;A TORLOKARAKTER COMMODORE-KODJA
;
        JMP COUT5
COUT2: CPI 0AH ;SOREMELES
        JNZ COUT3
;
        MVI A,17 ;A SOREMELES COMMODORE-KODJA
        JMP COUT5
;
COUT3: CPI 0DH ;KOCSI VISSZA
        JNZ COUT4
;
        CALL COUT5
        MVI A,145 ;EGY SOR AZ AUTO LF-HEZ
        JMP COUT5 ;KIMARAD
;
COUT4: CPI 20H ;RETURN HA KONTROL-KARAKTER
        RC ;
        CPI 80H ;RETURN, HA NEM ASCII-KARAKTER
        RNC ;
;
COUT5: STA DATA ;A KARAKTER ATVITELE A DATA-BA
        MVI A,VICOUT ;A KEPERNYOKIIRAS UTASITASA
        JMP I06510
;
LIST: ;EGY KARAKTER A C REGISZTERBOL A LIST-EGYSEGHEZ
        LDA IOTYPE ;MELYIK NYOMTATO
        ANI 04H ;0,HA 1515-OS, 1,HA 4022-ES
        MOV A,C ;EGY KARAKTER AZ A REGISZTERBE
        JNZ LIST2 ;UGRAS, HA NINCS CSERE
;
        LDA IOTYPE
        ANI 08H ;MELYIK TIPUSRA KELL VALTANI
        MOV A,C ;EGY KARAKTER BETOLTESE
        JNZ LIST1

```

```

4550 :
4560 : ACDA CDF4AC
4570 : ACDD C3E3AC
4580 :
4590 : ACE0 CD08AD
4600 : ACE3 3201F9
4610 : ACE6 3E05
4620 : ACE8 C394AB
4630 :
4640 :
4650 : ACEB 3E04
4660 : ACED CD94AB
4670 : ACF0 3A01F9
4680 : ACF3 C9
4690 :
4700 :
4710 : ACF4 FE41
4720 : ACF6 D8
4730 :
4740 : ACF7 FE5B
4750 : ACF9 DA05AD
4760 :
4770 : ACFC FE61
4780 : ACFE D8
4790 :
4800 : ACFF FEFB
4810 : AD01 D0
4820 :
4830 : AD02 E65F
4840 : AD04 C9
4850 :
4860 : AD05 F620
4870 : AD07 C9
4880 :
4890 : AD08 FE41
4900 : AD0A D8
4910 : AD0B FE60
4920 : AD0D D213AD
4930 :
4940 : AD10 F680
4950 : AD12 C9
4960 :
4970 : AD13 E65F
4980 : AD15 C9
4990 :
5000 :
5010 : AD16 79
5020 : AD17 00
5030 : AD18 C9
5040 :
5050 :
5060 : AD19 3E1A
5070 : AD1B E67F
5080 : AD1D C9
5090 :
5100 :
5110 :
5120 :
5130 :
5140 :
5150 :
5160 : 0400 =
5170 : 0100 =
5180 : 0011 =
5190 : 0002 =
5200 : 0022 =
5210 : 0001 =

;
CALL SWAP ;NAGY/KISBETUK FELCSERELESE
JMP LIST2

;
LIST1: CALL SWAP2 ;A 4022-ES CSERERUTINJA
LIST2: STA DATA ;EGY KARAKTER ATADASA
MVI A,VICPRT ;AZ 1515-RE
LIST3: JMP ID6510

;
LISTST: ;A LIST EGYSEG STATUSZA (0,HA KESZ;EGYEBKENT 1)
MVI A,VICPRT ;A NYOMTATOSTATUSZ UTASITASA
CALL ID6510
LDA DATA ;A STATUSZ: DATA
RET

;
SWAP: ;A NAGY/KISBETU CSEREJE A COMMODORE 64-EN
CPI 41H ;KISEBB, MINT A NAGY 'A'
RC ;RETURN,HA IGEN

;
CPI 5BH ;NAGYBETU
JC SWAP1 ;UGRAS,HA IGEN

;
CPI 61H ;KISEBB, MINT A KIS 'A'
RC ;UGRAS,HA IGEN

;
CPI 7BH ;KISBETU
RNC ;RETURN,HA NEM

;
ANI 5FH ;AZ 5. BIT TORLESE
RET

;
SWAP1: ORI 20H ;AZ 5. BIT BEALLITASA
RET

;
SWAP2: CPI 41H ;CY,HA KISEBB, MINT A NAGY 'A'
RC
CPI 60H ;CY,HA 40H < A < 60H
JNC SWAP3

;
ORI 80H
RET

;
SWAP3: ANI 5FH
RET

;
PUNCH ;EGY KARAKTER A C REGISZTERBOL A PUNCH EGYSEGRE
MOV A,C ;KARAKTER A REGISZTERBEN
NOP
RET ;NULLA RUTIN

;
READER: ;EGY KARAKTER A READER EGYSEGROL AZ A REGISZTERBE
MVI A,1AH ;FILE VEGE KARAKTER (CTRL Z) VISSZA!
ANI 7FH ;PARITASBIT TORLESE
RET

;
;
;*****
;*
;* KONSTANSOK A HOSTLEMEZHEZ
;*
;*
;*****
5160 : 0400 = BLKSIZ EQU 1024 ;A CP/M HELYIGENYE
5170 : 0100 = HSTSIZ EQU 256 ;A HOST LEMEZ SZEKTORNAGYSAGA
5180 : 0011 = HSTSPT EQU 17 ;HOST LEMEZ SZEKTOR/SAV
5190 : 0002 = HSTBLK EQU HSTSIZ/128 ;CP/M SZEKTOR/HOSTPUFFER
5200 : 0022 = CPMSPT EQU HSTBLK * HSTSPT ;CP/M SZEKTOR/SAV
5210 : 0001 = SECMSK EQU HSTBLK-1 ;SZEKTORMASZK

```



```

5220 : 0001 = SECDEF EQU 1 ; LOG2 (HSTBLK)
5230 :
5240 :
5250 :
5260 :
5270 :
5280 :
5290 : 0000 = WRALL EQU 0 ; WRITE TO ALLOCATED
5300 : 0001 = WRDIR EQU 1 ; WRITE TO DIRECTORY
5310 : 0002 = WRUAL EQU 2 ; WRITE TO UNALLOCATED
5320 :
5330 :
5340 : HOME: A KIVALASZTOTT LEMEZ HOME-MUVELETE
5350 : AD1E 3A11AF LDA HSTWRT ; VIZSGALAT WRITE-HOZ
5360 : AD21 B7 ORA A
5370 : AD22 C228AD JNZ HOMED
5380 : AD25 3210AF STA HSTACT ; HOST-AKTIV KAPCSOLO TORLESE
5390 : HOMED:
5400 : AD2B C9 RET
5410 :
5420 : SELDSK:
5430 : ; LEMEZVALASZTAS
5440 : AD29 210000 LXI H,0000H ; A VISSZAADOTT HIBAKOD
5450 : AD2C 79 MOV A,C ; A KIVALASZTOTT LEMEZ SZAMA
5460 : AD2D 3207AF STA SEKDSK ; A LEMEZSZAM TAROLASA
5470 : AD30 FE02 CPI 2 ; 0 VAGY 1 KELL HOGY LEGYEN
5480 : AD32 D0 RNC ; HA NINCS ATVITEL, EGGYEL NO
5490 : AD33 6F MOV L,A ; A LEMEZSZAM SZORZASA
5500 : AD34 29 DAD H ; 16-AL HL ALAPJAN
5510 : AD35 29 DAD H
5520 : AD36 29 DAD H
5530 : AD37 29 DAD H
5540 : AD38 1134AA LXI D,DPBASE ; A PARAMETERBLOKK BAZISA
5550 : AD3B 19 DAD D ; HL=DPB
5560 : AD3C C9 RET
5570 :
5580 : SETTRK:
5590 : ; A SAVSZAM BEALLITASA A BC REGISZTERBOL
5600 : AD3D 60 MOV H,B
5610 : AD3E 69 MOV L,C
5620 : AD3F 2208AF SHLD SEKTRK ; A KERESETT SAVHOZ
5630 : AD42 C9 RET
5640 :
5650 : SETSEC:
5660 : ; A SZEKTORSZAM BEALLITASA A C REGISZTERBOL
5670 : AD43 79 MOV A,C
5680 : AD44 320AAF STA SEKSEC ; A KERESETT SZEKTORHOZ
5690 : AD47 C9 RET
5700 :
5710 : SETDMA:
5720 : ; A DMA CIM BEALLITASA A BC REGISZTERBOL
5730 : AD48 60 MOV H,B
5740 : AD49 69 MOV L,C
5750 : AD4A 221BAF SHLD DMAADR
5760 : AD4D C9 RET
5770 :
5780 : SECTRAM:
5790 : ; A BC REGISZTERBELI SZEKTORSZAM
5800 : AD4E 60 MOV H,B
5810 : AD4F 69 MOV L,C
5820 : AD50 C9 RET
5830 :
5840 :
5850 :
5860 : A READ DEGRASI PONT KOVETI AZ
5870 : ELOZETES BIOS MEGHATAROZAST
5880 :
5890 :

```

```

5900 :
5910 :
5920 : AD51 AF
5930 : AD52 3212AF
5940 : AD55 3E01
5950 : AD57 3219AF
5960 : AD5A 3218AF
5970 : AD5D 3E02
5980 : AD5F 321AAF
5990 : AD62 C3D0AD
6000 :
6010 :
6020 :
6030 :
6040 :
6050 :
6060 :
6070 :
6080 :
6090 : AD65 AF
6100 : AD66 3219AF
6110 : AD69 79
6120 : AD6A 321AAF
6130 : AD6D FE02
6140 : AD6F C289AD
6150 :
6160 :
6170 : AD72 3E08
6180 : AD74 3212AF
6190 : AD77 3A07AF
6200 : AD7A 3213AF
6210 : AD7D 2A08AF
6220 : AD80 2214AF
6230 : AD83 3A0AAF
6240 : AD86 3216AF
6250 :
6260 :
6270 :
6280 : AD89 3A12AF
6290 : AD8C B7
6300 : AD8D CACBAD
6310 :
6320 :
6330 : AD90 3D
6340 : AD91 3212AF
6350 : AD94 3A07AF
6360 : AD97 2113AF
6370 : AD9A BE
6380 : AD9B C2CBAD
6390 :
6400 :
6410 : AD9E 2114AF
6420 : ADA1 CD6BAE
6430 : ADA4 C2CBAD
6440 :
6450 :
6460 : ADA7 3A0AAF
6470 : ADAA 2116AF
6480 : ADAD BE
6490 : ADAE C2CBAD
6500 :
6510 :
6520 : ADB1 34
6530 : ADB2 7E
6540 : ADB3 FE22
6550 : ADB5 DAC1AD
6560 :

```

```

READ:
;A KIVALASZTOTT CP/M SZEKTOR OLVASASA
XRA A
STA UNACNT
MVI A,1
STA READOP ; READ MUVELET
STA RSFLAG ; AZ ADATOK OLVASASA
MVI A,WRUAL
STA WRTYPE ; LEKEZELES 'NEM FOGLALT'-KENT
JMP RWOPER ; A READ-UTASITAS VEGREHAJTASA

;
;*****
; *
; * A WRITE BEUGRASI PONT KOVETI AZ *
; * ELOZETES BIOS MEGHATAROZAST *
; *
;*****
WRITE:
;A KIVALASZTOTT SZEKTOR IRASA
XRA A ; 0 AZ AKKUBA
STA READOP ; NEM OLVASASI MUVELET
MOV A,C ; A WRITE TIPUSA C ALAPJAN
STA WRTYPE
CPI WRUAL ; WRITE NEM FOGLALT
JNZ CHKUNA ; SZABAD HELY KERESESE

;
; WRITE NEM FOGLALT, PARAMETER BEALLITASA
MVI A,BLSIZ/128 ; A KOVETKEZO SZABAD REKORD
STA UNACNT
LDA SEKDSK ; LEMEZ
STA UNADSK ; UNADSK = SEKDSK
LHLD SEKTRK
SHLD UNATRK ; UNATRK = SEKTRK
LDA SEKSEC
STA UNASEC ; UNASEC = SEKSEC

;
CHKUNA:
;SZABAD SZEKTOR KERESESE WRITE-HOZ
LDA UNACNT ; VAN-E MEG SZABAD HELY
ORA A
JZ ALLOC ; UGRAS,HA NINCSE

;
; TOVABBI SZABAD REKORDOK
DCR A ; UNACNT = UNACNT-1
STA UNACNT
LDA SEKDSK ; AZONOS LEMEZ
LXI H,UNADSK
CMP M ; SEKDSK = UNADSK
JNZ ALLOC ; UGRAS,HA NEM

;
; AZONOS LEMEZ
LXI H,UNATRK
CALL TRKCMP ; SEKTRK = UNATRK
JNZ ALLOC ; UGRAS,HA NEM

;
; AZONOS SAV
LDA SEKSEC ; AZONOS SZEKTOR
LXI H,UNASEC
CMP M ; SEKSEC = UNASEC
JNZ ALLOC ; UGRAS,HA NEM

;
; A KOVETKEZO SZEKTOR KESOBBI HOZZAFERESHEZ
INR M ; UNASEC = UNASEC+1
MOV A,M ; VEGE A SAVNAK
CPI CPMSPT ; CP/M SZEKTORSZAMLALO
JC NOOVF ; UGRAS,HA NINCSE ATVITEL

```

```

6570 : ; ATVITEL A KOVETKEZO SAVRA,
6580 : ADB8 3600 MVI M,0 ;UNASEC = 0
6590 : AD8A 2A14AF LHLD UNATRK
6600 : ADBD 23 INX H
6610 : ADBE 2214AF SHLD UNATRK ;UNATRK = UNATRK+1
6620 : ;
6630 : NOOVF:
6640 : ADC1 AF XRA A ;0 AZ AKKUBAN
6650 : ADC2 3218AF STA RSFLAG ;RSFLAG = 0
6660 : ADC5 C3D0AD JMP RWOPER ;IRASRA
6670 : ;
6680 : ALLOC:
6690 : ADC8 AF XRA A ;0 AZ AKKUBAN
6700 : ADC9 3212AF STA UNACNT ;UNACNT = 0
6710 : ADCC 3C INR A ;1 AZ AKKUBAN
6720 : ADCD 3218AF STA RSFLAG ;RSFLAG = 1
6730 : ;
6740 : ; *****
6750 : ; *
6760 : ; * A READ ES WRITE KOZOS KODJAI *
6770 : ; *
6780 : ; *****
6790 : RWOPER:
6800 : ;UGRAS A READ-RA ES A WRITE-RA
6810 : ADD0 AF XRA A ;AZ AKKU TORLESE
6820 : ADD1 3217AF STA ERFLAG ;NEM VOLT HIBA
6830 : ADD4 3A0AAF LDA SEKSEC ;A HOST SEKTOR KISZAMITASA
6840 : ADD7 B7 ORA A ;CARRY = 0
6850 : ADD8 1F RAR ;ELTOLAS JOBBRA
6860 : ADD9 320FAF STA SEKHST ;HOST SEKTOR
6870 : ;
6880 : ;
6890 : ADDC 2110AF LXI H,HSTACT;HOST-AKTIV KAPCSOLO
6900 : ADDF 7E MOV A,M
6910 : ADE0 3601 MVI M,1 ;MINDIG 1
6920 : ADE2 B7 ORA A ;VOLT MAR
6930 : ADE3 CA0AAE JZ FILHST ;A HOST FELTOLTESE,HA NEM
6940 : ;
6950 : ;
6960 : ADE6 3A07AF AZONOS A KERESO PUFFERREL
6970 : ADE9 210BAF LDA SEKDSK
6980 : ADEC BE LXI H,HSTDSK;AZONOS LEMEZ
6990 : ADED C203AE CMP M ;SEKDSK = HSTDSK
7000 : JNZ NONTCH
7010 : ;
7020 : ADF0 210CAF AZONOS LEMEZ, AZONOS SAV
7030 : ADF3 CD6BAE LXI H,HSTTRK
7040 : ADF6 C203AE CALL TRKCMP ;SEKTRK = HSTTRK
7050 : JNZ NONTCH
7060 : ;
7070 : ADF9 3A0FAF AZONOS LEMEZ, AZONOS SAV, AZONOS PUFFER
7080 : ADFC 210EAF LDA SEKHST
7090 : ADFF BE LXI H,HSTSEC;SEKHST = HSTSEC
7100 : AE00 CA27AE CMP M
7110 : JZ MATCH ;UGRAS,HA EGYEZIK
7120 : ;
7130 : NONTCH:
7140 : AE03 3A11AF JO LEMEZ,ROSSZ SEKTOR
7150 : AE06 B7 LDA HSTWRT ;A HOST TELEIRVA
7160 : AE07 C477AE ORA A
7170 : CNZ WRHST ;A HOSTPUFFER TORLESE
7180 : ;
7190 : FILHST:
7200 : ;A HOSTPUFFER TOLTESE
7210 : AE0A 3A07AF LDA SEKDSK
7220 : AE0D 320BAF STA HSTDSK
7230 : AE10 2A08AF LHLD SEKTRK
7230 : AE13 220CAF SHLD HSTDRK

```

```

7240 : AE16 3A0FAF      LDA SEKHST
7250 : AE19 320EAF      STA HSTSEC
7260 : AE1C 3A18AF      LDA RSFLAG ; READ
7270 : AE1F B7          ORA A
7280 : AE20 C4CAAE      CNZ RDHST ; IGEN
7290 : AE23 AF          XRA A ; AZ AKKU TORLESE
7300 : AE24 3211AF      STA HSTWRT ; NEM WRITE
7310 :
7320 : ;
7330 : MATCH:
7340 : AE27 3A0AAF      ; AZ ADATOK MASOLASA A PUFFERBE
7350 : AE2A E601          LDA SEKSEC ; A PUFFEREK SZAMA
7360 : AE2C 6F           ANI SETMSK ; A LEGALSO BIT LEVALASZTASA
7370 : AE2D 2600         MOV L,A ; A 2 BYTE-OS SZAMLAO
7380 : AE2F 29          MVI H,0 ; ELTOLASADZ
7390 : AE30 29          DAD H ; 7-TEL BALRA ELTOLNI
7400 : AE31 29          DAD H
7410 : AE32 29          DAD H
7420 : AE33 29          DAD H
7430 : AE34 29          DAD H
7440 : AE35 29          DAD H
7450 : AE36 1100FB      LXI D,HSTBUF
7460 : AE39 19          DAD D ; HL A HOSTCIM
7470 : AE3A EB          XCHG ; MOST A DE-BEN
7480 : AE3B 2A1BAF      LHL DMAADR ; ADATOK OLVASASA/IRASA A CP/M-BEN
7490 : AE3E 0E00         MVI C,128 ; AZ IRANYT
7500 : AE40 3A19AF      LDA READOP ; MEGHATARAZO SZAM
7510 : AE43 B7          ORA A
7520 : AE44 C24DAE      JNZ RNMVME ; UGRAS, HA OLVASAS
7530 :
7540 : ;
7550 : AE47 3E01          ; WRITE MUVELETEK TARTOLASA ES AZ IRANY MEGFORDITASA
7560 : AE49 3211AF      MVI A,1
7570 : AE4C EB          STA HSTWRT ; HSTWRT = 1
7580 :                  XCHG ; FORRAS/CEL FELCSERELESE
7590 : ;
7600 : RNMVME:
7610 :                  ; C 128-AT TARTALMAZ, DE A FORRAS, HL A CEL
7620 : AE4D 1A          LDAX D ; EGY KARAKTER A FORRASTERULETROL
7630 : AE4E 13          INX D
7640 : AE4F 77          MOV M,A ; A CELTERULETRE
7650 : AE50 23          INX H
7660 : AE51 0D          DCR C ; 128 SOR
7670 : ;
7680 : ;
7690 : AE55 3A1AAF      AZ ADATOK TOVABBITASA A HOSTPUFFERBOL/BA
7700 : AE58 FE01          LDA WRTYPE ; WRITE TIPUS
7710 : AE5A 3A17AF      CPI WRDIR ; A TARTALOMJEGYZEKBE
7720 : AE5D C0          LDA ERFLAG ; HA HIBA
7730 :                  RNZ ; NINCIS, TOVABBI FELDOLBOZAS
7740 : ;
7750 : AE5E B7          ; HOSTPUFFER TARTALOMJEGYZEKBE IRASADZ
7760 : AE5F C0          ORA A ; HIBA
7770 : AE60 AF          RNZ ; HA IGEN, KESZ
7780 : AE61 3211AF      XRA A ; AZ AKKU TORLESE
7790 : AE64 CD77AE      STA HSTWRT ; A PUFFER MEGTELT
7800 : AE6A C9          CALL WRHST
7810 :                  LDA ERFLAG
7820 :                  RET
7830 : ;
7840 : ; *****
7850 : ; * SEBEDRUTIN A 16-BITES OSSZEHASONLITASHOZ *
7860 : ; *
7870 : ; *****
7880 : TRKCMP:
7890 : ; HL = .UNATRK VAGY .HSTTRK OSSZEHASONLITASA A
7900 : AE6B EB          ; SEKTRK-VAL
7910 :                  XCHG

```

```

7910 : AE6C 210BAF      LXI  H,STKTRK
7920 : AE6F 1A          LDAX  D      ;LO BYTE OSSZEHASONLITASA
7930 : AE70 BE          CMP  M      ;EGYENLO
7940 : AE71 C0          RNZ      ;RETURN HA NEM
7950 :                  ; LO BYTE EGYENLO; A HI BYTE VIZSGALATA
7960 : AE72 13          INX  D
7970 : AE73 23          INX  H
7980 : AE74 1A          LDAX  D
7990 : AE75 BE          CMP  M      ;A KAPCSOLOK BEALLITASA
8000 : AE76 C9          RET
8010 :
8020 :
8030 :
8040 :
8050 :
8060 :
8070 :
8080 :
8090 :
8100 :
8110 :
8120 :
8130 : AE77 3E01
8140 : AE79 321DAF      WRHST0: STA  RW      ;BEIRASA AZ UTASITASREGISZTERBE
8150 : AE7C 3A0BAF      LDA  HSTDSK ;A HOSTLEMEZSZAM BEOLVASASA
8160 : AE7F 3204F9      STA  DISKNO ;ES BEALLITASA
8170 : AE82 CDA5AE      CALL CHGDSK ;VALODI VIRTUALIS LEMEZ
8180 : AE85 3A0CAF      WRHST2: LDA  HSTTRK ;A HOST SAVSZAM BETOLTESE
8190 : AE88 3C          INR  A      ;ES EGY HOZZAADASA
8200 : AE89 FE12          CPI  18    ;RELATIV CIMKENT (OFFSET-KENT)
8210 : AE8B DABFAE      JC   WRHST3 ;CARRY,HA A SAV < MINT 18
8220 : AE8E 3C          INR  A
8230 : AE8F 3203F9      WRHST3: STA  TRACK ;IRAS A KOZOS TERULETRE
8240 : AE92 3A0EAF      LDA  HSTSEC ;A HOST SEKTORSZAM BEOLVASASA
8250 : AE95 3202F9      STA  SECTOR ;ES IRASA A KOZOS TERULETRE
8260 : AE98 3A1DAF      LDA  RW      ;LEMEZMUVELET
8270 : AE9B CD94AB      CALL 106510
8280 : AE9E 3A01F9      LDA  DATA ;A LEMEZ-STATUSZ BEOLVASASA
8290 : AEA1 3217AF      STA  ERFLAG ;ES AZ ERFLAG TARTOLASA
8300 : AEA4 C9          RET
8310 :
8320 : AEA5 67
8330 : AEA6 3AFFFC      CHGDSK: MOV  M,A    ;A LEMEZSZAM TARTOLASA
8340 : AEA9 E6B1      LDA  IOTYPE ;BIT 0-0 VIRTUALIS EBETEM
8350 : AEB3 DB          ANI  01
8360 : AEB6 3204F9      RNZ      ;NEM 0,HA KETTO EGYSEG VAN
8370 : AEBF 7C          STA  DISKNO ;AZ 'A' EGYSEG
8380 :                  MOV  A,H    ;LEMEZSZAMNAK BETOLTESE
8390 :
8400 : AEB0 211EAF      LXI  H,CURDSK;A MI LEMEZUNK
8410 : AEB3 BE          CMP  M
8420 : AEB4 DB          RZ      ;RETURN,HA IGEN !
8430 :
8440 : AEB5 77
8450 : AEB6 C641
8460 : AEB8 32DDAE      MOV  M,A    ;AZ UJ LEMEZSZAM BEALLITASA
8470 : AEBB 21CFAE      ADI  'A'    ;AZ ASCII-KARAKTER ELOALLITASA
8480 : AEBE CDFAAE      STA  DISKINT ;ES ELHELVEZESE AZ UZENETBEN
8490 : AEC1 CD0AAC
8500 : AEC6 C2C1AE
8510 : AEC9 C9
8520 :
8530 :
8540 :
8550 :
8560 :
8570 : AECA 3E00

```

WRHST: ;HSTDSK = HOSTLEMEZ #, HSTTRK = HOSTSAV #,
 ;HSTSEC = HOSTSEKTOR #. FELIR "HSTSIZ" BYTE-OT
 ;A HSTBUF-BOL ES A HIBAT VISSZAADJA AZ ERFLAG-BAN
 ;ERFLAG NEM 0, HA HIBA TORTENT
 MVI A,VICWR ;A DISK-WRITE UTASITAS
 WRHST0: STA RW ;BEIRASA AZ UTASITASREGISZTERBE
 LDA HSTDSK ;A HOSTLEMEZSZAM BEOLVASASA
 STA DISKNO ;ES BEALLITASA
 CALL CHGDSK ;VALODI VIRTUALIS LEMEZ
 WRHST2: LDA HSTTRK ;A HOST SAVSZAM BETOLTESE
 INR A ;ES EGY HOZZAADASA
 CPI 18 ;RELATIV CIMKENT (OFFSET-KENT)
 JC WRHST3 ;CARRY,HA A SAV < MINT 18
 INR A
 WRHST3: STA TRACK ;IRAS A KOZOS TERULETRE
 LDA HSTSEC ;A HOST SEKTORSZAM BEOLVASASA
 STA SECTOR ;ES IRASA A KOZOS TERULETRE
 LDA RW ;LEMEZMUVELET
 CALL 106510
 LDA DATA ;A LEMEZ-STATUSZ BEOLVASASA
 STA ERFLAG ;ES AZ ERFLAG TARTOLASA
 RET
 ;
 CHGDSK: MOV M,A ;A LEMEZSZAM TARTOLASA
 LDA IOTYPE ;BIT 0-0 VIRTUALIS EBETEM
 ANI 01
 RNZ ;NEM 0,HA KETTO EGYSEG VAN
 STA DISKNO ;AZ 'A' EGYSEG
 MOV A,H ;LEMEZSZAMNAK BETOLTESE
 ;
 LXI H,CURDSK;A MI LEMEZUNK
 CMP M
 RZ ;RETURN,HA IGEN !
 ;
 MOV M,A ;AZ UJ LEMEZSZAM BEALLITASA
 ADI 'A' ;AZ ASCII-KARAKTER ELOALLITASA
 STA DISKINT ;ES ELHELVEZESE AZ UZENETBEN
 ;
 LXI H,MNTHSG; 'INSERT DISK' UZENET
 CALL PMSG ;KIIRASA ES
 CALL CONIN ;VARAKOZAS A 'RETURN'-RE
 JNZ CHGD1
 RET
 ;
 RDHST: ;HSTDSK = HOSTLEMEZ #, HSTTRK = HOSTSAV #,
 ;HSTSEC = HOSTSEKTOR #. "HSHSIZ" BYTE OLVASASA
 ;A HIBAKAPCSOLO BEALLITASA AZ ERFLAG SZERINT
 MVI A,VICRD ;DISK-READ UTASITAS

```

8580 : AECC C379AE          JMP WRHST0 ;A TOBBI UGYANAZ MINT A WRITE-NAL
8590 :                      ;
8600 : AECF 0D0A496E73MNTMSG: DB 0DH,0AH,'INSERT DISK'
8610 : AEDD 41          DSKMNT: DB 'A'
8620 : AEDE 20696E746F      DB ' INTO DRIVE 0, PRESS RETURN '
8630 : AEF9 00          DB 00H
8640 :                      ;
8650 : AEFA 7E          MSG: MOV A,M
8660 : AEFB A7          ANA A
8670 : AEFC C8          RZ
8680 : AEFD E5          PUSH H
8690 : AEFE 4F          MOV C,A
8700 : AEFF CDB3AC      CALL CONOUT
8710 : AF02 E1          POP H
8720 : AF03 23          INX H
8730 : AF04 C3FAAE      JMP MSG
8740 :                      ;
8750 :                      ;*****
8760 :                      ;*
8770 :                      ;*      NEM INICIALIZALT RAM ADATTERULET      ;*
8780 :                      ;*
8790 :                      ;*****
8800 :                      ;
8810 : AF07          SEKDSK: DS 1          ;SEEK LEMEZSZAM
8820 : AF08          SEKTRK: DS 2          ;SEEK SAVSZAM
8830 : AF0A          SEKSEC: DS 1          ;SEEK SZEKTORSZAM
8840 :                      ;
8850 : AF0B          HSTD SK: DS 1          ;HOST LEMEZSZAM
8860 : AF0C          HSTTRK: DS 2          ;HOST SAVSZAM
8870 : AF0E          HSTSEC: DS 1          ;HOST SZEKTORSZAM
8880 :                      ;
8890 : AF0F          SEKHST: DS 1          ;SEEK SHIFT JOBBRA SECSHF
8900 : AF10          HSTACT: DS 1          ;HOST AKTIV-KAPCSOLO
8910 : AF11          HSTWRT: DS 1          ;HOST IRASKAPCSOLO
8920 :                      ;
8930 : AF12          UNACNT: DS 1          ;UNALLOC REKORDSZAMLALO
8940 : AF13          UNADSK: DS 1          ;AZ UTOLSO UNALLOC LEMEZ
8950 : AF14          UNATRK: DS 2          ;AZ UTOLSO UNALLOC SAV
8960 : AF16          UNASEC: DS 1          ;AZ UTOLSO UNALLOC SZEKTOR
8970 :                      ;
8980 : AF17          ERFLAG: DS 1          ;HIBAKAPCSOLO
8990 : AF18          RSFLAG: DS 1          ;READ-SZEKTOR KAPCSOLO
9000 : AF19          RADOP: DS 1          ;1 READ MUVELET
9010 : AF1A          WRTYPE: DS 1          ;WRITE MUVELET
9020 : AF1B          DMAADR: DS 2          ;AZ UTOLSO DMA-CIM
9030 : AF1D          RW: DS 1          ;ATMENETI UTASITASREGISZTER
9040 : AF1E          CURDSK: DS 1          ;VIRTUALIS LEMEZMUTATO
9050 :                      ;
9060 :                      ;      RAM A BDOS SZAMARA
9070 : AF1F =          BEGDAT EQU $          ;AZ ADATTERULET KEZDETE
9080 : AF1F          DIRBF: DS 128          ;A TARTALOMJEGYZEK
9090 : AF9F          ALL00: DS 31          ;ALLOCATION VEKTOR 0
9100 : AFBE          ALL01: DS 31          ;ALLOCATION VEKTOR 1
9110 : AFDD          CHK00: DS 16          ;CHECK VEKTOR 0
9120 : AFED          CHK01: DS 16          ;CHECK VEKTOR 1
9130 :                      ;
9140 : AFFD =          ENDDAT EQU $          ;AZ ADATTERULET VEGE
9150 : 00DE =          DATSIZ EQU $-BEGDAT ;AZ ADATTERULET MERETE
9160 : AFFD

```

6.8 Saját INPUT/OUTPUT rutin beépítése a BIOS-ba

Az előző fejezetben közölt BIOS listából kiderül, hogy a PUNCH és a READER rutinokhoz létezik ugyan ugróvektor, de a tényleges rutinok egyetlen RETURN utasításból állnak. A READER rutin híváskor egy CTRL-Z karaktert továbbít, mintha elértük volna a feldolgozandó szöveg végét.

Semmilyen elvi akadály nincs azonban annak, hogy a rendszert saját READER rutinnal bővítsük.

Hasonló a helyzet a PUNCH rutin esetében, ez is csak elvileg létezik, de felhasználhatjuk ezt a lehetőséget pl. egy interface-szel támogatott Centronics nyomtató illesztésére (l. a 7.1 fejezetben).

Mivel a 7-es és 8-as utasításkódok a \$E00, ill. \$F00 6510-es ugrási címeket használják, a saját rutinjaikat a 6510-es kódrendszerben kell megírni.

A megírt rutint pl. a következőképpen hívhatjuk:

```
PUNCH: ;Egy karakter továbbítása a PUN-hoz: a Centronics nyomtatóra
        MOV A, C      ;Egy karakter továbbítása a PUN-hoz: a Centronics nyomtatóra
        STA DATA     ;az átviteli regiszterbe
        MVI A,7       ;A saját rutin kódja
        STA CMD
        CALL IO6510   ;A 6510-es rutin hívása
        RET
```

A 6510-es output rutint a \$E00-ás kezdőcímtől kell elhelyezni, ahol 256 byte áll rendelkezésre, de ez biztosan elegendő is, hiszen a rutin feladata mindössze annyi, hogy az adatokat a 6510-es porthoz továbbítsa.

A saját READER rutin hívása:

```
READER: ;Egy karakter az RDR-ről
        MVI A,8       ;A saját rutin utasításkódja
        STA CMD
        CALL IO6510   ;A 6510-es rutin hívása
        LDA DATA     ;A karakter beolvasása
        ANI 7FH       ;A paritásbit törlése
        RET
```

A 8-as utasításkód alapján a rendszer az input rutint keresi, ahol szintén 256 byte-tal gazdálkodhatunk. Tetszőleges külső egységet (kazettás-, lemezegység stb.) hozzárendelhetünk a RDR rendszerperifériához, sőt, ha V24-es interface-t használunk, megoldhatjuk az alapgépek közötti adatforgalmat is. A READER rutin a továbbítandó szöveget ASCII kódban várja, és a szöveg végén egy CTRL-Z karaktert továbbít, általában ugyanis a programok (pl. a PIP) a CTRL-Z-t tekintik a feldolgozás végét jelző karakternek.

6.9 Adattorgalom a CP/M programok és a Commodore BASIC között

A CP/M operációs rendszer programjai által előállított szöveges file-okat, ill. adatokat általában nem használhatjuk a Commodore BASIC programok inputjaként. A BIOS 65 megfelelő részét módosítva azonban ezeket az egyébként csak CP/M módban elérhető file-okat a BASIC programok számára is hozzáférhetővé tehetjük.

A CP/M rendszerben létrehozott file-okat kinyomtathatjuk a BIOS 65 megfelelő OPEN és PRINT# utasításaival. Ha az OPEN utasításban az egységszámot megváltoztatjuk, az adatkiírás nem nyomtatóra, hanem az általunk kijelölt egységre, pl. a kazettára vonatkozhat.

Ahhoz, hogy a rendszer a szalag file-t megnyissa, 1-es másodlagos címet kell megadni. A DDT programmal módosíthatjuk a CP/M-et:

DDT

– SFAC7

FAC7 07 01

FAC8 20

– SFADD

FADD 00 01

FADE A9

– SFAE6

FAE6 04 01

FAE7 20

– AC

Módosítás után a AP vagy a PIP LST:= adatkiírások már nem a nyomtatót, hanem a kazettás magnetofont tekintik output egységnek. A képernyőn megjelenik a 'press record play on tape' üzenet és a képernyő színe sötétre vált. A teljes szöveg kiírása után nyomjuk le a CTRL-C-t, majd melegstart közben a STOP és RESTORE billentyűket egyidejűleg. A gép ekkor a READY üzenettel Commodore üzemmódban jelentkezik vissza. A CLOSE 4 utasítással zárjuk le a szalag file-t, majd a gép ki-/bekapcsolása után BASIC programmal olvassuk vissza.

100 OPEN1

110 GET# 1, A\$

...

200 IF ST < > 64 THEN 110

210 CLOSE1

A karakteresen beolvasott adatokat (A\$) Commodore üzemmódban igény szerint feldolgozhatjuk.

Ha a CP/M file-okat gyakran használjuk Commodore üzemmódban, célszerű erre a célra programot írni, hogy ne kelljen minden esetben újra módosítani a rendszert. A programot az ED szövegszerkesztővel és az ASM assemblerrel készíthetjük el:

```
ED - SZALAG.ASM
```

```
ORG    100H
MVI     A,1
STA     0FAC7H
STA     0FADFH
STA     0FAE6H
JMP     0
```

A következő utasítással fordítsuk le a programot:

```
ASM SZALAG.AAX
```

Készítsünk belőle egy .COM file-t

```
LOAD SZALAG
```

A továbbiakban a rendszermódosítást bármikor megtehetjük a SZALAG nevű program segítségével.

A fordított irányú adatátvitelhez, Commodore üzemmódból CP/M üzemmódba, egy kis trükkre van szükség. A szövegből vagy programból egy \$1100 betöltési címmel ellátott program-file-t kell létrehoznunk.

A betöltési cím a Z80-as 100H, amely a TPA (Transient Program Area – átmeneti programterület) kezdőcíme.

A következő BASIC program egy soros file-ból elkészíti a megfelelő programfile-t:

```
0 REM **** P59. ****
1 :
2 :
100 INPUT "A FILE NEVE: ";N$
110 OPEN 2,8,2,N$
120 OPEN 1,8,1,N$+".CPM":REM A FILE MEGNYITASA
130 PRINT#1,CHR$(0)CHR$(17);:REM KEZDOCIIM
140 GET#2,A$:IF ST=64 THEN CLOSE 2:CLOSE 1:END
150 PRINT#1,A$:GOTO 140
KESZ .
```

A program betöltése előtt meg kell néznünk a program méretét (a tartalomjegyzék betöltésével):

```
25 NEV.CPM PRG
```

A LOAD "NEV.CPM",8,1 paranccsal töltjük be a programot, majd a szokásos módon helyezzük be a CP/M lemezt és indítjuk el a rendszert.

A SAVE 25 NEV.TXT utasítással tárolhatjuk az eredeti BASIC szöveget CP/M üzemmódban. A 25-ös szám meghatározza a tároláshoz szükséges 256 byte-os blokkok számát. Tárolás után a kapott file-ból CP/M rendszer alatt feldolgozhatjuk az adatokat.

Ez az eljárás egy esetben okozhat problémát, ha a szövegfile kis/nagybetűs üzemmódban készült. Ekkor a tároló BASIC program 140-es sorát meg kell változtatni; a beolvasott karakter helyett a megfelelő ASCII kódot kell az 1-es file-ba írni.

7. FEJEZET

ILLESZTÉSI ÉS BŐVÍTÉSI LEHETŐSÉGEK A COMMODORE 64-ESEN

7.1 Centronics nyomtató illesztése a számítógéphez

A Commodore 64-esbe beépítettek egy olyan hardveres illesztőt, amelyet az operációs rendszer nem használ. Angol elnevezése (user port) is arra utal, hogy a gép használói szabadon rendelkezhetnek alkalmazásáról. A user port egy nyolc bites és két ún. handshake (kézfogó) vezetékből áll. A nyolc bit mindegyike külön kapcsolható, és input/output műveletekhez egyaránt alkalmazható. Az alábbiakban bemutatjuk, hogy miként csatlakoztatható egy Centronics nyomtató az alapgéphez a user porton keresztül.

A folyamat röviden a következő:

A byte-ot bitenként nyolc párhuzamos adatvezeték szállítja. Két handshake vezeték biztosítja, hogy az átvitel során adatok ne vesszenek el. Mielőtt a gép egy byte-ot a nyomtatóhoz továbbítana, megvizsgálja, hogy a nyomtató készen áll-e az adat fogadására. Ezt a vizsgálatot a BUSY (foglalt) vezeték végzi el. Ha a nyomtató készenléti állapotban van, a gép az adatot a porthoz továbbítja, és ezt a tényt a STROBE vezetéken át közli a nyomtatóval. A nyomtató átveszi az adatot, és mindaddig, míg fel nem dolgozta (tárolás, nyomtatás), a BUSY vezeték bitjét magasan tartja. Feldolgozás után megkezdődhet a következő adatbyte továbbítása, és a folyamat addig ismétlődik, míg a gép be nem fejezi az adatok küldését.

Hogyan programozható az interface?

Ahhoz, hogy PRINT# utasítással a nyomtatóhoz adatokat küldhessünk, az adatátvitelt lebonyolító szoftvernek az operációs rendszerben kell elhelyezkednie. Első adódik néhány probléma:

A legtöbb Centronics nyomtató pl. ASCII karakterkészlettel dolgozik, ami nem azonos a Commodore 64-es karakterkészletével. Ahhoz tehát, hogy a nyomtató felismerje a hozzáküldött karaktereket, konvertálnunk kell a gép kódrendszerét a nyomtató kódrendszerére. Ráadásul időnként arra is szükség lehet, hogy a Commodore grafikusjeleit is felismerje a nyomtató, amelyek egyáltalán nem szerepelnek a karakterkészletében. Ezeknek a problémáknak a megoldására készítettünk egy olyan vezérlő programot, amelyet kétféleképpen használhatunk. Ha az OPEN utasításban nem adunk meg másodlagos címet, akkor az átvitel során a Commodore kódokat a program ASCII kódokká alakítja. Ha megadjuk a másodlagos címet, az adatok változatlan formában kerülnek át a géptől a nyomtatóra. Egységszámként 2-est adunk meg. Ezt az egységszámot általában az RS232-es interface foglalja le, de mivel az általunk készített program is a user portot használja, a kettő együtt semmiképpen nem működhet, tehát a 2-es egységszám szabad.

Egy program kinyomtatása a következő:

```
OPEN 1,2 : CMD 1 : LIST
```

Miután a kurzort visszakaptuk

```
PRINT# 1 : CLOSE1
```

utasítással zárjuk le a CMD üzemmódot és a csatornát. Ha grafikus karaktereket akarunk nyomtatni, az utasítás a következő lehet:

```
OPEN 1,2,1
PRINT # 1,...
CLOSE1
```

```
OPEN 1,2,1
CLOSE 1,2,1
```

A nyomtató hardveres illesztéséhez mindössze egy olyan kábelre van szükség, amelynek egyik vége a user porthoz, másik vége pedig a Centronics nyomtatóhoz csatlakozik. (A vezetékek tűállítását a programlista végén megadtuk.) Ezzel a vezetékekkel csatlakoztassuk a nyomtatót a géphez; kapcsoljuk be először a gépet, majd a nyomtatót! Töltsük be a gépi kódú programot, majd SYS 12*4096 utasítással hajtuk végre.

Az alábbiakban közöljük a gépi kódú programot és BASIC betöltőjét.

```
0 REM **** P60. ****
1 :
2 :
100 : ;CENTRONICS-INTERFACE A CBM64-EN
110 : ;
120 : ;ILLESZTES AZ 'USER-PORT'-RA
130 : ;
140 : ;AZ I/O VEKTOROK MEGHATAROZASA
150 : ;
200 : 031A ;== $31A
210 : 031C OPENV == **2
220 : 031E CLOSEV == **2
230 : 0320 CHKINV == **2
240 : 0322 CHKOUTV == **2
260 : 0326 == **4
270 : 0328 BSOUTV == **2
280 : 0328 XREG = $97 ;A REGISZTER TARCIME
281 : ;
282 : ;A PORT MEGHATAROZASA
283 : ;
284 : DD00 ;== $DD00 ;CIA2
285 : DD01 PORTA == **1 ;PORT
286 : DD02 PORTB == **1
287 : DD03 DRRA == **1 ;ADATIRANY
288 : DD0D DRRB == **10
289 : DD0E IRC == **1 ;MEGSZAKITAST VEZERLO REGISZTER
290 : DD0E LF = $B8
291 : DD0E SA = $B9
292 : DD0E FA = $BA
293 : DD0E NMBFILES = $98
294 : DD0E LFTAB = $259
295 : DD0E FATAB = $263
296 : DD0E SATAB = $26D
297 : DD0E SRCHFIL = $F30F
300 : ;
310 : ;INICIALIZALAS
320 : ;
330 : C000 INIT == $C000
340 : C000 A9 58 LDA #<OPEN
350 : C002 A0 C0 LDY #>OPEN
360 : C004 8D 1A 03 STA OPENV
370 : C007 8C 1B 03 STY OPENV+1
380 : C00A A9 8B LDA #<CLOSE
390 : C00C A0 C0 LDY #>CLOSE
400 : C00E 8D 1C 03 STA CLOSEV
410 : C011 8C 1D 03 STY CLOSEV+1
420 : C014 A9 A3 LDA #<CHKIN
430 : C016 A0 C0 LDY #>CHKIN
```

```

440 : C018 8D 1E 03 STA CHKINV
450 : C01B 8C 1F 03 STY CHKINV+1
460 : C01E A9 BA LDA #< CHKOUT
470 : C020 A0 C0 LDY #> CHKOUT
480 : C022 8D 20 03 STA CHKOUTV
490 : C025 8C 21 03 STY CHKOUTV+1
500 : C028 A9 D1 LDA #< BSOUT
510 : C02A A0 C0 LDY #> BSOUT
520 : C02C 8D 26 03 STA BSOUTV
530 : C02F 8C 27 03 STY BSOUTV+1
540 : C032 A9 FF LDA #FF
550 : C034 8D 03 DD STA DRRD ;KIIRAS A B PORT-ON
560 : C037 AD 02 DD LDA DRRR
570 : C03A 09 04 ORA #100
580 : C03C 8D 02 DD STA DRRR ;KIIRAS A PAZ-N
590 : C03F 60 RTS
591 :
592 : ;KIIRAS HANDSHAKE-EL
593 : ;ADATOK A B PORT-RA
594 : ;A STROBE A PAZ-ON
595 : ;A BUSY AZ ICR SZERINT
596 : ;
600 : C040 8D 01 DD AUSGABE STA PORTB ;AZ ADATOK KIIRASA
610 : C043 A9 10 LDA #10000 ;A FLAG-KAPCSOLO BIT-MASZKJA
620 : C045 2C 0D DD TESTBUSY BIT ICR
630 : C048 F0 FB BEQ TESTBUSY
640 : C04A AD 00 DD LDA PORTA
650 : C04D 09 04 ORA #100 ;A STROBE BEALLITASA
660 : C04F 8D 0D DD STA PORTA
670 : C052 29 FB AND #11111011 ;A STROBE TORLESE
680 : C054 8D 00 DD STA PORTA
690 : C057 60 RTS
695 :
700 : C058 A6 B8 OPEN LDX LF ;LOGIKAI FILE-SZAM
710 : C05A F0 05 BEQ OPENERR
720 : C05C 20 0F F3 JSR SRCHFIL ;FILE-SZAM KERESSESE
730 : C05F D0 03 BNE **5
740 : C061 4C FE F6 OPENERR JMP $F6FE ;'FILE OPEN ERROR'
750 : C064 A6 98 LDX NMBFILES ;MEGVITOTT FILE-K SZAMA
760 : C066 E0 0A CPX #10
770 : C068 90 03 BCC **5
780 : C06A 4C FB F6 JMP $F6FB ;'TOO MANY FILES ERROR'
790 : C06D E6 98 INC NMBFILES
800 : C06F A5 B8 LDA LF
810 : C071 9D 59 02 STA LFTAB,X
820 : C074 A5 B9 LDA SA
830 : C076 09 60 ORA #60
840 : C078 9D 6D 02 STA SATAB,X
850 : C07B A5 BA LDA FA
860 : C07D 9D 63 02 STA FATAB,X
870 : C080 C9 02 CMP #2
880 : C082 D0 02 BNE **4
890 : C084 18 CLC
900 : C085 60 RTS ;KESZ
910 : C086 C9 00 CMP #0
920 : C088 4C 77 F3 JMP $F377 ;TOVABB
925 :
930 : C08B 20 14 F3 CLOSE JSR $F314 ;LOGIKAI FILE-SZAM KERESSESE
940 : C08E F0 02 BEQ **4
950 : C090 18 CLC
960 : C091 60 RTS ;KESZ
970 : C092 20 1F F3 JRS $F31F ;A FILE-PARAMETER BEALLITASA
980 : C095 BA TXA
990 : C096 48 PHA
1000 : C097 A5 BA LDA FA
1010 : C099 C9 02 CMP #2
1020 : C09B F0 03 BEQ **5

```

```

1030 : C09D 4C 9D F2      JMP $F29D ;TOVABB
1040 : C0A0 4C F1 F2      JMP $F2F1 ;A FILE-BEJEGYZES TORLESE
1045 :
1050 : C0A3 20 0F F3 CHKIN JSR SRCHFIL ;A FILE-SZAM KERESSESE
1060 : C0A6 F0 03      BEQ **5
1070 : C0AB 4C 01 F7      JMP $F701 ;'FILE NOT OPEN ERROR'
1080 : C0AE 20 1F F3      JSR $F31F ;A FILE-PARAMETER BEALLITASA
1090 : C0AE A5 BA      LDA FA
1100 : C0B0 C9 02      CMP #2
1110 : C0B2 D0 03      BNE **5
1120 : C0B4 4C 0A F7      JMP $F70A ;'NOT INPUT FILE ERROR'
1125 :
1130 : C0B7 4C 19 F2      JMP $F219
1140 : C0BA 20 0F F3 CHKOUT JSR SRCHFIL ;FILE-SZAM KERESSESE
1150 : C0BD F0 03      BEQ **5
1160 : C0BF 4C 01 F7      JMP $F701 ;'FILE NOT OPEN ERROR'
1170 : C0C2 20 1F F3      JSR $F31F ;FILE-PARAMETER BEALLITASA
1180 : C0C5 A5 BA      LDA FA
1190 : C0C7 C9 02      CMP #2
1200 : C0C9 D0 03      BNE **5
1210 : C0CB 4C 75 F2      JMP $F275
1220 : C0CE 4C 5B F2      JMP $F25B
1225 :
1230 : C0D1 4B          ; BSOUT PHA
1240 : C0D2 A5 9A      LDA $9A ;KIMENETI EGYSEG
1250 : C0D4 C9 02      CMP #2
1260 : C0D6 F0 03      BEQ **5
1270 : C0D8 4C CD F1      JMP $F1CD ;TOVABB
1280 : C0DB A5 B9      LDA 9A ;HASODLAGOS CIM
1290 : C0DD 29 0F      AND #%1111
1300 : C0DF D0 0A      BNE OUT ;NEM EGYENLO NULLAVAL
1310 : C0E1 86 97      STX XREG
1320 : C0E3 68          PLA
1330 : C0E4 AA          TAX
1340 : C0E5 BD F3 C0     LDA TABLE,X ;A KOD BETOLTESE A TABLAZATBOL
1350 : C0E8 A6 97      LDX XREG
1360 : C0EA 24          .BYT $24
1370 : C0EB 68          PLA
1380 : C0EC 4B          PHA
1390 : C0ED 20 40 C0     JSR AUSGABE ;KIIRAS;EGY KARAKTER KIIRASA
1400 : C0F0 68          PLA
1410 : C0F1 18          CLC
1420 : C0F2 60          RTS
1430 : C0F3
1440 : C0F3          TABLAZAT= *
1440 : C0F3 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
1450 : C103 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
1460 : C113 20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F
1470 : C123 30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F
1480 : C133 40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F
1490 : C143 50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F
1500 : C153 60 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F
1510 : C163 70 71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F
1520 : C173 80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F
1530 : C183 90 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F
1540 : C193 A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB AC AD AE AF
1550 : C1A3 B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB BC BD BE BF
1560 : C1B3 C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF
1570 : C1C3 D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 DA DB DC DD DE DF
1580 : C1D3 E0 E1 E2 E3 E4 E5 E6 E7 E8 E9 EA EB EC ED EE EF
1590 : C1E3 F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB FC FD FE FF

```

A BASIC betöltőprogram:

```

0 REM **** P61. ****
1 :
2 :
100 FOR I=49152 TO 49650
110 READ X:POKE I,X:S=S+X:NEXT
120 DATA 169, 88,160,192,141, 26, 3,140, 27, 3,169,139
130 DATA 160,192,141, 28, 3,140, 29, 3,169,163,160,192
140 DATA 141, 30, 3,140, 31, 3,169,186,160,192,141, 32
150 DATA 3,140, 33, 3,169,209,160,192,141, 38, 3,140
160 DATA 39, 3,169,255,141, 3,221,173, 2,221, 9, 4
170 DATA 141, 2,221, 96,141, 1,221,169, 16, 44, 13,221
180 DATA 240,251,173, 0,221, 9, 4,141, 0,221, 41,251
190 DATA 141, 0,221, 96,166,184,240, 5, 32, 15,243,208
200 DATA 3, 76,254,246,166,152,224, 10,144, 3, 76,251
210 DATA 246,230,152,165,184,157, 89, 2,165,185, 9, 96
220 DATA 157,109, 2,165,186,157, 99, 2,201, 2,208, 2
230 DATA 24, 96,201, 0, 76,119,243, 32, 20,243,240, 2
240 DATA 24, 96, 32, 31,243,138, 72,165,186,201, 2,240
250 DATA 3, 76,157,242, 76,241,242, 32, 15,243,240, 3
260 DATA 76, 1,247, 32, 31,243,165,186,201, 2,208, 3
270 DATA 76, 10,247, 76, 25,242, 32, 15,243,240, 3, 76
280 DATA 1,247, 32, 31,243,165,186,201, 2,208, 3, 76
290 DATA 117,242, 76, 91,242, 72,165,154,201, 2,240, 3
300 DATA 76,205,241,165,185, 41, 15,208, 10,134,151,104
310 DATA 170,189,243,192,166,151, 36,104, 72, 32, 64,192
320 DATA 184, 24, 96, 0, 1, 2, 3, 4, 5, 6, 7, 8
330 DATA 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20
340 DATA 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32
350 DATA 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44
360 DATA 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56
370 DATA 57, 58, 59, 60, 61, 62, 63, 64, 97, 98, 99,100
380 DATA 101,102,103,104,105,106,107,108,109,110,111,112
390 DATA 113,114,115,116,117,118,119,120,121,122,123,124
400 DATA 125, 94, 95, 96, 97, 98, 99,100,101,102,103,104
410 DATA 105,106,107,108,109,110,111,112,113,114,115,116
420 DATA 117,118,119,120,121,122,123,124,125,126,127,128
430 DATA 129,130,131,132,133,134,135,136,137,138,139,140
440 DATA 141,142,143,144,145,146,147,148,149,150,151,152
450 DATA 153,154,155,156,157,158,159,160,161,162,163,164
460 DATA 165,166,167,168,169,170,171,172,173,174,175,176
470 DATA 177,178,179,180,181,182,183,184,185,186,187,188
480 DATA 189,190,191,192, 65, 66, 67, 68, 69, 70, 71, 72
490 DATA 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84
500 DATA 85, 86, 87, 88, 89, 90, 91, 92, 93,222,223,224
510 DATA 225,226,227,228,229,230,231,232,233,234,235,236
520 DATA 237,238,239,240,241,242,243,244,245,246,247,248
530 DATA 249,250,251,252,253,254,255
540 IF S<>58502 THEN PRINT "HIBA A DATA SORBAN..!":END
550 PRINT "OK..!!"
KESZ .

```

A kábelkapcsolat a *nyomtató* és a *user port* között a következőképpen néz ki:

| USER PORT | — | CENTRONICS |
|--------------|--------------|------------|
| A | GND | 16 |
| B | FLAG - BUSY | 11 |
| C | D0 | 2 |
| D | D1 | 3 |
| E | D2 | 4 |
| F | D3 | 5 |
| H | D4 | 6 |
| J | D5 | 7 |
| K | D6 | 8 |
| L | D7 | 9 |
| M | PA2 - STROBE | 1 |

7.2 A számítógépek közötti adatátvitel

Tegyük fel, hogy az Olvasónak van egy Commodore 64-es és egy CBM 8032-es gépe. Szeretné az egyik gazdag grafikus lehetőségeit a másik adattároló kapacitásával együtt érvényesíteni. Miért ne lehetne a 8032-esről érkező adatokat egy, a Commodore 64 grafikus lehetőségeit tartalmazó programmal feldolgozni! Vagy megfordítva; esetleg valaki a Commodore 64-esről érkező adatokat a 8032-es 80 oszlopos kijelzési lehetőségével szeretné megjeleníteni. Miért ne?

Most bemutatunk egy olyan programot, amely lebonyolítja a két számítógép közötti adatforgalmat. A Commodore 64-es partnere ez esetben egy CBM 8032-es számítógép, de lehetne tetszőleges külső egység, pl. Centronics nyomtató is. A program újabb példa a user port célszerű alkalmazására.

Természetesen a két egységet hardveresen össze kell kapcsolni. Az összekötő vezeték túállását a programlista után közöljük.

A programban felhasznált változók:

- X Az érkező vagy továbbítandó karakter ASCII kódja
T1 Rendszeróra. Üteme: 1/60 s.
D\$ Az érkező vagy továbbítandó teljes fűzér.

A CBM 8032-es érintett regiszterei:

- 59457 A user port adatregisztere
59459 A user port adatirányregisztere. Mint tudjuk, a user porton át egyaránt küldhetünk és fogadhatunk adatokat. Ezért minden bitre megadható az adatáramlás iránya.
59468 Ennek a címnek az 5. bitje vezérli a user port CB2-es vezetéket. Továbbító üzemmódban az adatok érvényességét jelzi, fogadó üzemmódban pedig nyugtázza az adatok vételét. Ezzel kizárjuk az adatvesztés lehetőségét.
59469 A cím 1. bitje visszaadja a CA1 vezetékek állapotát. Továbbító üzemmódban biztosítja a várakozást a visszaigazolásra, fogadó üzemmódban az adatküldő „érvényes” üzenetére.

A Commodore 64-es tárcímek:

- 59576 A cím 2. bitje vezérli a user port PA2-es vezetéket. Feladata azonos az 59468 címnél leírtakkal
59577 A user port adatregisztere
59579 A user port adatirány-regisztere
59589 A cím 4. bitje visszaadja a user port FLAG vezetékének állapotát
Feladata azonos az 59469 címnél leírtakkal.

A program dokumentációja:

- 1000–1080 Továbbító rutin
1000 Az adatirány-regisztert beállítjuk adatok küldésére
1010 A továbbító ciklus hossza azonos az átvitelre kerülő byte-ok számával
1020 Az I-edik byte-ot (D\$ tartalmát) beírjuk az adatregiszterbe
1030 Az adat érvényességét 0 értékkel jelezzük
1040 Várakozó ciklus a fogadó egység visszaigazolására
1050–1060 Az érvényességet jelző érték ismét 1, és folytatódik az adatátvitel, ha még van továbbítandó adat.
1070–1080 A user port nyugalmi állapotba kerül, és kilépünk a továbbító rutinból
2000–2090 Fogadó rutin
2000 Várakozás az első byte-ra
2020–2030 Az óra értékét 0-ra állítjuk, és/vagy 2 másodperc várakozás, vagy a soron következő byte fogadása zajlik, vagy vége az adatátvitelnek. Az időhatárt a 2020-as sorban megváltoztathatjuk. A mi példánkban a 2020-as sorban lekérdezzük, hogy a TI több mint 120 időegységet tartalmaz-e, ami pontosan 2 s-nak felel meg ($120 \cdot \frac{1}{60}$ s). A 3 s-nak 180 időegység felelne meg.
2040–2050 Várakozás az érvényességi üzenetre.
Ha az adat érvényes, az érkező byte-ot hozzáfűzzük a D\$ változó eddigi tartalmához.
2060–2080 A visszaigazolást jelző vezeték értékét ismét 0-ra állítjuk, majd várakozunk a következő karakterre.

A rutinok felépítése nagyon egyszerű. Ha rendelkezésre áll a megfelelő kábel, és a két gépet összekapcsoltuk, minden megy a maga útján.

A program gyakorlatilag két alapvető részből áll: a fogadó és a továbbító rutinból, amelyeket más programokban is felhasználhatunk. Ha az egységnek karaktereket akarunk küldeni; helyezzük el azokat a D\$ változóban, majd hívjuk az 1000-es rutint. Ha adatokat fogadunk, hívjuk meg a 2000-es rutint, majd az érkező karaktert fűzzük hozzá a D\$ változó eddigi tartalmához. Az alábbiakban két programlistát közlünk, egyet a C-64-es, egyet pedig a CBM 8032-es gépre. A programok szerkezetileg hasonlóak, kivéve természetesen a user port címét. Még egy különbséget találunk a 2010-es sorban, ennek magyarázata is a user port eltérő kezelésében rejlik.

```
0 REM **** F62. ****
1 :
2 :
995 REM ADATATVITEL AZ 'USER-PORT'-ON, SZUBRUTINGYUJTEMENY
996 REM 6522-ES VALTOZAT A CBM8032-ES 59456-OS TARCIMEN
998 REM
999 REM EGY FUZER TOVABBITASA
1000 POKE 59459,255:REM ADATIRANY KIMENET
1005 POKE 59468,PEEK(59468) OR 224:REM CB2 HIGH
1010 FOR I=1 TO LEN(D$):REM D$ TOVABBITASARA SZOLGALO CIKLUS
1020 X=ASC(MID$(D$,I,1)):POKE 59457,X:REM ADATOK KIIRASA
1030 POKE 59468,PEEK(59468) AND 223:REM CB2 LOW
1040 WAIT 59469,2:REM VARAKOZS AZ ADATOKRA
1050 POKE 59468,PEEK(59468) OR 224:REM CB2 HIGH
1060 NEXT
1070 POKE 59457,0:POKE 59459,0:REM A PORT VISSZAALLITASA
```

```

1080 RETURN
1090 REM
1100 REM A D$ FOGADASA
1110 REM
1120 REM
2000 WAIT 59469,2:REM VARAKOZAS AZ ADATATVITEL MEGKEZDESERE
2010 D$="":REM A D$ ELOKESZITESE
2020 TI$="000000"
2030 IF TI$>120 THEN 2090
2040 IF (PEEK(59469) AND 2)=0 THEN 2030:REM VARAKOZAS AZ ADATBYTE-RA
2050 X=PEEK(59457):D$=D$+CHR$(X)
2060 POKE 59468,PEEK(59468) AND 223:REM CB2 LOW
2070 POKE 59468,PEEK(59468) OR 224:REM A FOGADO ALLAPOTA
2080 GOTO 2020
2090 RETURN

0 REM **** P62/1. ****
1 :
2 :
995 REM AZ ADATATVITEL 'USER-PORT'-ON,SZUBRUTINGYUJITEMENY
996 REM A 6526-OS VALTOZAT A CBM64-ES 56576-OS TARCIMEN
998 REM
999 REM EGY FUZER TOVABBITASA
1000 POKE 56579,255:REM ADATIRANY KIMENET
1010 FOR I=1 TO LEN(D$):REM A D$ TOVABBITASARA SZOLGALO CIKLUS
1020 X=ASC(MID$(D$,I,1)):POKE 56577,X:REM ADATOK KIIRASA
1030 POKE 56576,147:REM PA2 LOW
1040 WAIT 56589,16:REM VARAKOZAS AZ ADATOKRA
1050 POKE 56576,151:REM PA2 HIGH
1060 NEXT
1070 POKE 56577,0:POKE 56579,0:REM A PORT VISSZAALLITASA
1080 RETURN
1090 REM
1100 REM A D$ FOGADASA
1110 REM
1120 REM
2000 WAIT 56589,16:REM VARAKOZAS AZ ADATATVITEL MEGKEZDESERE
2010 D$="":GOTO 2050:REM A D$ ELOKESZITESE
2020 TI$="000000"
2030 IF TI$>120 THEN 2090
2040 IF (PEEK(56589) AND 16)=0 THEN 2030:REM VARAKOZAS AZ ADATBYTE-RA
2050 X=PEEK(56577):D$=D$+CHR$(X)
2060 POKE 56576,147:REM PA2 LOW
2070 POKE 56576,151:REM PA2 HIGH
2080 GOTO 2020
2090 RETURN

```

Nézzünk egy példát a rutinok felhasználásának megvilágítására. Ha az összekapcsolt gépek-be a megfelelő programokat betöltöttük, írjuk be a küldő egységen a

```
10 D$ = "TESZT": GUSUB 1000: END
```

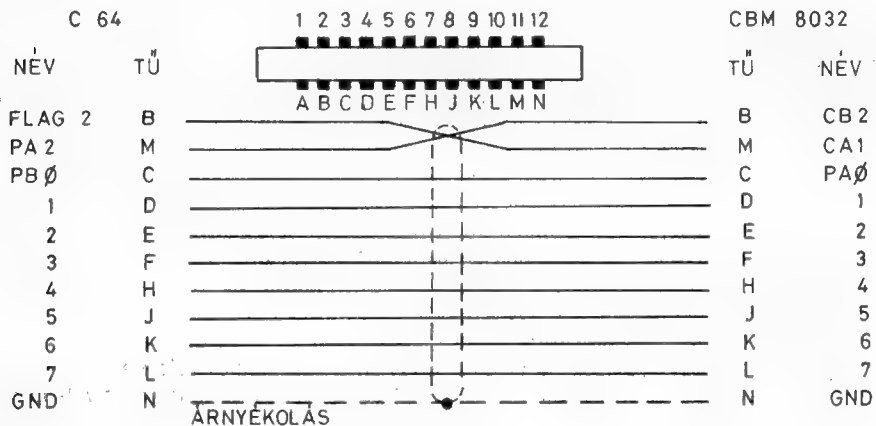
a fogadó egységen pedig a

```
10 GOSUB 2000: PRINT D$: END
```

programsort.

Indítsuk el a programokat, és nézzük meg mi történik. Az összekötő vezetékét az alábbi ábra szemlélteti. Célszerű kábelként egy 10 eres (egyenként 0,14–0,35 mm² keresztmetszetű) árnyékol-t vezetékét használni. Az árnyékolást mindkét oldalon csatlakoztassuk a GND tűhöz. Az

általunk végzett kísérletnél a rendszer 5 méteres kábellel hibátlanul működött. Hogy ennél nagyságrenddel hosszabb vezeték esetében sem lép fel hiba, azt nem állíthatjuk, ugyanis ekkor a környezet zavaró hatását nem lehet kizárni (pl. elektromotor működése, mosógép, porszívó stb.). Három méteres kábel mellett az átvitel teljesen biztonságos.



7.3 A CP/M cartridge illesztése az expansion porthoz

Esettanulmány

Ebben a fejezetben bemutatjuk, hogy hogyan lehet a Commodore 64-eshez kiegészítő hardvert illeszteni az expansion porton keresztül. A kiegészítő hardver jelen esetben egy CP/M cartridge, amelyet a Commodore cég fejlesztett ki.

Mi is a CP/M cartridge?

A CP/M operációs rendszer hardveres változata, amely tartalmaz egy Z80-as mikroproceszort és egy olyan logikai egységet, amely megteremti a processzorok közötti kommunikációs lehetőséget. Illesztésével a Commodore 64-esen futtathatók a CP/M szoftvertermékek.

Az alábbiakban ismertetjük a két processzor hardveres összekapcsolását, a műveleti egységek működését és a cartridge vázlatos kapcsolási rajzát. A magyarázatok megértéséhez az Olvasónak nem kell hardveres szakembernek lennie.

A legfontosabb szerepet az expansion port vezetékei töltik be:

| | |
|--------|---|
| CD0–7 | A rendszerbusz adatvezetékei Ezeket a C64-es 6510-es processzora csak addig vezérelheti, amíg a DMA = 1 és a BA = 1. Ez a továbbiakban igen fontos tény. |
| CA0–15 | A rendszerbusz címvezetéke. Erre ugyanaz érvényes, mint az előzőre. |
| I/O1 | A vezetékek értéke 0, ha a \$DE00 – \$DEFF (56382–57087) tártartományban valamilyen művelet zajlik. |
| RES | Ha értéke 0 (általában bekapcsoláskor), akkor minden hardveregység alapállapotba kerül. |
| DMA | Bemeneti vezetékek. Ha értéke 0, a 6510-es felfüggeszti a működését, átadva a rendszerbuszt a külső vezérlésnek. |
| BA | Ezen a vezetéken jelzi a C64-es videovezérlő, hogy hozzáfér a tárhoz (BA = 0). Ez idő alatt a rendszerbuszt sem a 6510-es, sem a port nem használhatja. |
| S02 | A rendszerütem, amely a C64-es műveleteit összehangolja. Az együttműködés érdekében ez az ütem érvényes a CP/M Z80-as processzorára is. |

Kezdjük a folyamatok leírását a nyugalmi, azaz bekapcsolás utáni állapottal. A Z80-as BUSREQ vezetékek ugyanazt a szerepet töltik be, amit az AEC vezetékek a 6510-es processzornál, azaz ha értéke 0, a Z80-as felfüggeszti a munkát, és felszabadítja a rendszerbuszt. Amikor a készüléket bekapcsoljuk, a RES vezetékek értéke egy pillanatra 0 lesz, és az FF flip-flop visszabillen eredeti állapotába ($Q = 0$, $Q = -1$).

Ezzel a BA értékétől függetlenül az ÉS kapu kimenete 0 lesz. Ennek az a következménye, hogy az A1, A2 és D pufferek lezáródnak, és a rendszerbusz vezérlése megszűnik. A BUSREQ = 0, tehát a Z80-as nyugalmi állapotban marad. Az elmondottakból kitűnik, hogy a rendszer tevékenységét az FF flip-flop és a BA jel (logikai ÉS kapcsolattal!) együttesen határozzák meg. Ha az FF értéke magas ($Q = 1$, $-Q = 0$) és ugyanakkor BA = 1, a BUSREQ értéke is 1 lesz, és így a Z80-as működésbe lép. Ettől kezdve a C64-es a megszokott módon dolgozik, mindaddig, amíg végre nem hajtunk egy POKE 56832,1 utasítást. Ahogyan az alábbi blokk-sémából látjuk, ez az utasítás aktivizálja az I/O1 vezetéket, ettől az FF flip-flop átbillen, és mivel a BA = 1, a Z80-as működni kezd. Ezzel egyidejűleg a DMA = 0 miatt a 6510-es kikapcsol, és a C64-es működésképtelen lesz, hiszen nincs a tárban olyan rendszerprogram, amit a Z80-as képes lenne helyesen végrehajtani.

Hová kell elhelyezni a Z80-as programokat? A 6510-essel ellentétben a Z80-as indításakor (RES = 0) a programok futtatását a 0-ás tárcímen kezdi. Az első konfliktust tehát az okozza, hogy a 0-ás címen a 6510-es I/O port található, vagyis olyan paraméterek, amelyek a rendszer

működéséhez feltétlenül szükségesek. Ugyanakkor a Z80-as ezen a tárterületen futtatható programot keres, és nem várhatjuk tőle, hogy más tárcímhez forduljon. Ezt a problémát a CP/M modul nagyon elegánsan oldja meg. A blokksmémán látunk egy ADD-vel jelölt műveleti egységet, amelyben van egy négy bites összeadó. Ez az egység két négy bites bemenettel rendelkezik, és ezek összegét továbbítja a kimenetre. Az összeadó kimenete a négy legnagyobb helyiértékű címbíthez csatlakozik. Az egyik bemenetére érkeznek a Z80-as címvezetékei, a másik bemenete konstans 1. Így a kimeneten a négy felső címbít 1-gyel növelt értéke adódik minden műveletnél.

Az egész (2 byte-os) címet ez a művelet \$1000-rel (4096-tal) növeli. Ha tehát a példánknak megfelelően a Z80-as kiadja a 0-s startcímet, azaz ott keresi az első végrehajtható utasítást, az összeadási művelet elvégzése után a tényleges startcím \$1000 lesz. Erre a területre viszont nyugodtan elhelyezhetjük a Z80-as rutinokat, hiszen ezt a területet a 6510-es nem használja. A fentiekből az is kiderül, hogy a \$F000 (61440) Z80-as cím tulajdonképpen a 0-ás címnek felel meg, hiszen a négybites összeadás az átvitt nem veszi figyelembe.

Mindezek alapján nézzük meg, mi történik bekapcsolás után! Betöltődik egy startprogram (\$1000-tól), az FF flip-flop átbillen, és a Z80-as működni kezd, azaz elindítja a \$1000 címen elhelyezett programot, ami többnyire egy CP/M betöltőprogram.

Betöltés után a rendszer fogadja és végrehajtja a használt utasításait. Természetesen a fentiek ismeretében saját programokat is elhelyezhetünk a \$1000-es címen, és így a Z80-as processzort tetszőleges, általunk kijelölt feladatok végrehajtására kényszeríthetjük.

Általában az ilyen programok megírásához szükség van a C64-es hardvercímeinek pontos ismeretére, hiszen a programok többsége input/output műveleteket végez, amelyek valamilyen külső egységre hivatkoznak. Nem szabad közben elfeledkeznünk az offset (viszonylagos) címzésről sem. Ha például az adatokat a user portra akarjuk továbbítani, a Z80-as programban a \$CD01 (52481) címet kell kijelölni, mivel a 6510-es processzor user port címe \$DD01 (56577). A különbség mindig \$1000 (4096)!

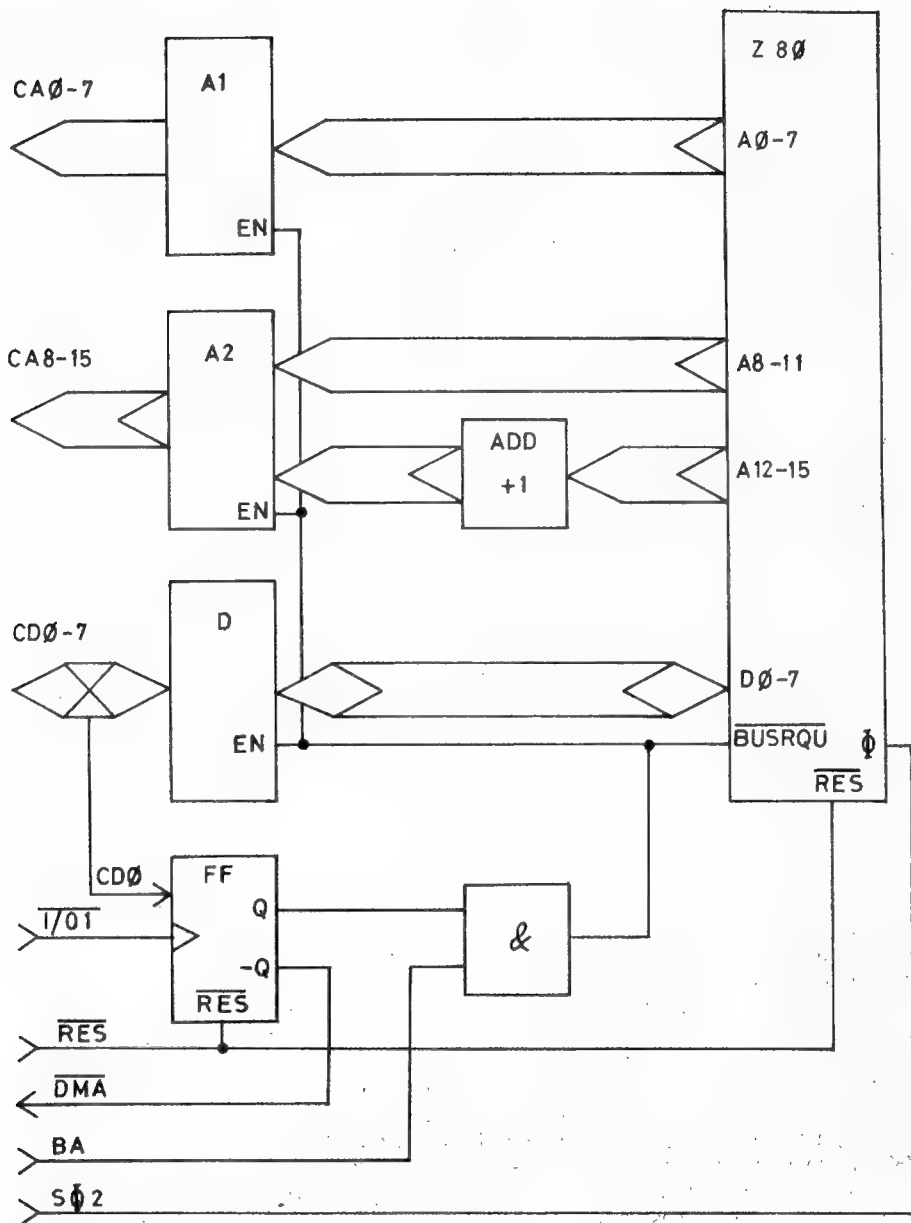
Hogyan térhetünk vissza a Z80-as üzemmódból? Nagyon egyszerűen. Ha a BASIC interpretert betöltöttük, adjuk ki a POKE 52736,0 utasítást az FF visszabillentéséhez. Ekkor a BUSREQ értéke 0 lesz, így a Z80-as felfüggeszti a működését és a DMA = 1 miatt a 6510-es bekapcsol és ott folytatja a munkát, ahol abbahagyta. A POKE utasításban szereplő címen természetesen a valódi címnél most is 4096-tal kevesebbnek kell lennie, hiszen az ADD művelet hozzáadja ezt az értéket. A POKE utasítás végrehajtásával még semmit sem oldottunk meg, hiszen most a 6510-es nem talál a tárban végrehajtható programot, így bár bekapcsol, mégis működésképtelen.

Térjünk ki még néhány szóban a BA vezetékek szerepére. A BA olyan segédjel, amely az adatbusz forgalmát szabályozza. Értéke meghatározó a CP/M-re nézve, hiszen logikai ÉS kapcsolatban van az FF flip-floppal, tehát az utóbbi értékétől függetlenül a Z80-as abbahagyja a működését, ha BA = 0.

A BA vezetékek fontos szerepet töltenek be a vezérlésben. Értékét a C64-es videovezérlője állítja be. Miután a képernyőtár a szokásos helyén maradt, és azt a videovezérlő ciklikusan felfrissíti, ez idő alatt a rendszerbuszt senki sem használhatja. Ha a gép a megszokott rendszerben dolgozik, a rendszerbuszt folyamatosan használja, tehát a videovezérlőnek ügyesen kell használnia azokat a pillanatokat, amikor a rendszerbusz szabad. (Ha sprite-okkal dolgozunk, még a szokásos üzemmódban is előfordulhat időzavar.)

A Z80-ast azonban, minthogy használja a rendszerbuszt, értesíteni kell arról, hogy a videovezérlő működési ideje alatt függessze fel a munkát. Erre a célra szolgál a BA vezetékek.

A Z80-as és a 6510-es processzorok interaktív munkáját szándékosan leegyszerűsítve ábrázoltuk. A folyamat lényegét az ábra alapján tökéletesen meg lehet érteni, a részletezés pedig csak zavarná az Olvasót.



7.4 Szintetizátor sztereóban

Aki gyakran használja a C64-esbe beépített szintetizátort, bizonyosan sokat bosszankodott már a tv-készülék rossz hangminőségén.

Ahhoz, hogy jobb hangminőséget kapjunk, egy jó minőségű sztereóberendezést kell csatlakoztatnunk a C64-eshez. A csatlakoztatás nem olyan egyszerű, ugyanis a szintetizátor egy db kimenetét fel kellene osztanunk a sztereóberendezés két csatornájára, holott a hangjeleket a szintetizátor együtt kezeli. Osszuk fel a frekvenciatartományt két részre, úgy, hogy az alsó és felső tartomány határa a 300 Hz-es frekvencia legyen. Ezzel tulajdonképpen a szintetizátor által átfogott hallható sávot középen választjuk szét, nevezetesen három alsó (36 Hz-ig terjedő) és négy felső (4800 Hz-ig terjedő) oktávra.

A felosztást két kettős T-szűrővel oldjuk meg, amelynek karakterisztika-meredeksége 6 dB/oktáv, határfrekvenciája pedig 300 Hz (aluláteresztő szűrő), ill. 3 kHz (feluláteresztő szűrő).

A határfrekvenciát megfelelő kondenzátorral ki-ki ízlése szerint megváltoztathatja, ha közben az ellenállás értékét nem változtatja meg, ami a csatlakoztatott készülék impedanciájától függ.

Egy adott F határfrekvenciához a szükséges kondenzátor értékét (C) a $C = 1/(3300 * F)$, ill. megfordítva az $F = 1/(3300 * C)$ képlet alapján számíthatjuk ki.

A gyártó cég laboratóriumában végzett kísérletek alapján, az általunk megadott adatok optimális hangminőséget biztosítanak.

A 3 dB/oktáv karakterisztika-meredekségű szűrő mellett nincs szükség az R2, C2, C4, R6-os szerkezeti elemekre, a hangszórók közötti akusztikai tartomány még így is széles lesz.

A szűrőket egy kis darab lyukraszteres lemezre építsük, majd csatlakoztassuk a sztereóberendezéshez. Ha a lemezt a kábelvégekkel együtt beöntjük szilikonkaucsukba, esztétikailag is kifogástalan megoldáshoz jutunk.

Most bemutatunk egy olyan programot, amely háromszöghullám formájú, „suhanó” hangot hoz létre, miközben kiválóan lehet hallani, hogyan vándorol a hang az egyik hangszóróból a másikba.

Összetettebb, felhangban gazdagabb hangzatoknál, például fűrészfog hullámformával az egyik hangszóróban a felhangok, a másikon pedig az alaphangok hallhatóak (300 Hz alatt), és a hangzás lényegesen terjedelmesebb lesz.

```
0 REM **** P63. ****
```

```
1 :
```

```
2 :
```

```
10 S1=53272
```

```
20 S2=54279
```

```
30 S3=54286
```

```
60 RS=54295
```

```
70 PL=54296
```

```
80 POKE S1+4,0:POKE S2+4,0:POKE S3+4,0
```

```
100 A=9:D=9:S=9:R=9:H=30
```

```
110 POKE RS,0:POKE PL,15
```

```
120 POKE S3+5,16*A+D:POKE S3+6,16*S+R
```

```
130 POKE S3+4,17
```

```
140 FOR I=0 TO H:POKE S3+1,PEEK(54300):NEXT I
```

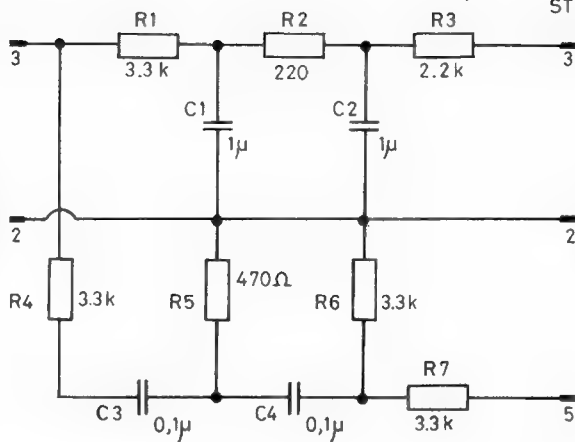
```
150 POKE S3+4,16
```

```
160 FOR I=0 TO R*4:POKE S3+1,PEEK(54300):NEXT I
```

Az alábbi kapcsolási rajzot a megértés érdekében leegyszerűsítettük. A sztereó-oldal dugós csatlakozóját a phono bemenethez terveztük.

C 64

STÉREO



8. FEJEZET

ADATFELDOLGOZÁS

Bevezetés

A legtöbb számítástechnikai feladatban későbbi feldolgozásra bizonyos adatokat tárolni kell. A programozás legnehezebb része általában a rugalmas, célszerű adatkezelés.

Teljesen mindegy, hogy milyen programnyelven dolgozunk (BASIC, FORTRAN, PASCAL stb.), az adatkezelés minden nyelven egyaránt fontos szerepet játszik.

Ebben a fejezetben betekintést nyújtunk a C64-es adatfeldolgozási adottságaiba. Az elméleti leírásokat mindvégig gyakorlati feladatokkal illusztráljuk, bízva abban, hogy ez elősegíti a könnyebb megértést. Először ismertetjük az adatfeldolgozás legfontosabb alapfogalmait.

FILE

Manapság az adatfeldolgozási szakkifejezések (file, rekord, mező stb.) szinte hétköznapi fogalmakká váltak, de még mindig nincs mindenki tisztában a jelentésükkel. Mit takar a „file” fogalom? Magyarozatként nézzünk egy példát: legtöbb intézményben a kézi nyilvántartásra kartonrendszert dolgoznak ki. A könyvtárban a leltárba vett könyvekről egy-egy kartont készítenek, amelyen a könyv összes lényeges adata szerepel. Általában a karton-rendszerű nyilvántartásokban a nyilvántartott tárgyak, személyek legfontosabb ismertetőjegyeit, azonosító adatait tüntetik fel. A könyvtári példánál maradva, a könyv kartonjára felvezetik pl., hogy az adott könyv melyik kölcsönzőnél van és mennyi ideig.

A kartonokat meghatározott rendszer szerint tárolják, leggyakrabban abc-sorrendben, esetleg leltári szám szerint, vagy a leltárbavétel dátuma szerint. A könyvtári példában az egyes könyvek adatait tartalmazó, adott sorrendben tárolt kartonok összessége a tulajdonképpeni katalógus.

A file a katalógussal analóg fogalom, amely a kartonokhoz hasonló egységekből, ún. rekordokból áll, és az egyes rekordok éppúgy, mint a kartonok, az egyedekre vonatkozó elemi információkat tartalmazzák.

A file rendkívüli előnye a kartonrendszerrel szemben a hihetetlen rugalmasság, a kis helyigény és a rendkívül gyors kezelhetőség. Ma már a mikroszámítógépek tárhatalma is meghaladja az 1 millió karaktert. Vajon hány kartonra lenne szükség ekkora információtömeg nyilvántartására?

REKORD

A rekord fogalma a fentiek alapján már világos. A kartonrendszer számítógépes megfelelője a file, a karton megfelelője a rekord. Az adott nyilvántartás minden egyedének megfelel egy rekord, amely az adott egyedre vonatkozó összes elemi információt tartalmazza. A rekord elemi részei a mezők.

MEZŐ

Angolul: field. Lassan kiderül, hogy a file, a rekord és a mező fogalmakat csak együtt tudjuk értelmezni. Visszatérve a kartonokra, a kartonokon szereplő egyedi (egyres) információit a számítógépesek adatmezőnek nevezik. A három új fogalmat a következőképpen szemléltethetjük:

FILE→REKORD→MEZŐ

Ha lentől felfelé építkezünk, azt mondhatjuk, hogy a rekordot összetartozó mezők, a file-t összetartozó rekordok alkotják.

Tekintsünk most egy olyan nyilvántartást, amelyben egy munkahely dolgozóinak nevét és lakcímét tároljuk:

FILE : CIMFILE

| | MEZO vezetéknév | MEZO keresztnev | MEZO város | MEZO utca | MEZO házszám |
|-----------|--------------------|--------------------|---------------|--------------|-----------------|
| 1. rekord | Valentinyi | Zoltán | Gödöllő | Kossuth | 12 |
| 2. rekord | Polgár | Szilárd | Göd | Vas | 13 |
| 3. rekord | Turcsányi | Attila | Üllő | Szép | 14 |

A címfile egy egysége az egy adott személy adatait tartalmazó rekord. Ebben a nyilvántartásban minden rekord öt mezőből áll, az első mező pl. a vezetéknév.

Az alapfogalmak tisztázása után betekintést nyújtunk az adatkezelés programozásába, illetve az adattárolás különböző módszereit ismertetjük.

8.1 Kazettás egység – lemezegység

A Commodore 64-es gépen az adatok tárolására két lehetőség kínálkozik: adattárolás kazettás egységen, ill. adattárolás lemezegységen.

Mi a különbség a két háttértároló között? Mikor melyiket válasszuk? Hogy ezekre a kérdésekre választ kapjunk, tegyünk egy kirándulást az időben visszafelé, az adattfeldolgozás kezdeti időszakára.

Néhány évtizeddel ezelőtt a „floppy” vagy „mágneslemez” fogalmak teljesen ismeretlenek voltak. Abban az időben az emberek nem találtak olyan eszközt, amelyen információt tárolhattak volna úgy, hogy az bármikor visszanyerhető legyen. (Eltekintve természetesen az információ szöveges vagy szimbolikus tárolásától.)

Ezt az eszközt ma általánosan adathordozónak nevezhetjük. Az első mai értelemben vett adathordozó a lyukkártya volt. Ez a viszonylag olcsó megoldás azzal az óriási hátránnyal járt, hogy bonyolult berendezéseket (nevezetesen kártyalyukasztót, ill. kártyaolvasót) kellett készíteni az információ tárolására és visszanyerésére. A berendezések mechanikus elven és ebből adódóan meglehetősen lassan működtek. Minthogy az idő előrehaladtával mind biztonságosabb és gyorsabb adattárolásra volt szükség, a szakemberek újabb megoldásokat kerestek. Az eredmény a mágnesszalagos adattárolás, amely ma a személyi számítógépeken (kazettás formában) és a nagyszámítógépes rendszereken egyaránt használatos, és az alapelv mindkét rendszernél ugyanaz, csak a méret különböző.

A mágnesszalagos adattárolás elve rendkívül egyszerű. A Commodore 64-eshez csatlakoztatott külső egységek egy azonosító számmal vannak ellátva. A kazettás egység száma az 1. Először nézzük meg, hogy lehet az adatokat a szalagra felírni. Írás előtt a következő utasítással egy file-t kell megnyitni a szalagon:

OPEN 1, 1, 1, "CBM-64 FILE"

Az első 1-es a C64-es számára megadott file-szám. Ha ugyanis egyszerre több file-lal dolgozunk, azokat egy ún. logikai file-számmal (1-től 255-ig) meg kell különböztetnünk egymástól.

Ha a kazettás egységen megnyomjuk a RECORD és a PLAY gombot, a Commodore 64 fejlécezt ír fel a szalagra. Ha egy file-t írunk fel a szalagra, a fejléc csak a file nevét, ha programot írunk fel, a program nevén kívül a kezdőcímét is tartalmazza. Ez az ún. program-, ill. file-fej kétszer egymás után kerül fel a szalagra. A fej felírása után a szalag megáll, majd következik a program vagy file tárolása. Ha az OPEN utasításban 2-es másodlagos címet használunk, a rendszer a file neve mögé EOT (End of tape = szalag vége) jelet ír, ami a Commodore 64-esnél azt jelenti, hogy a szalagra már nem lehet tovább írni:

OPEN 1, 1, 2, "CBM 64 FILE"

A szalagra felírt file-t a következő utasítással nyithatjuk meg olvasásra:

OPEN 1, 1, 0, "CBM 64 FILE"

Az utasítás hatására a Commodore 64 mindaddig keresi a file nevét a szalagon, míg meg nem találja, vagy míg szalag vége (EOT) jelet nem érzékel.

FONTOS: a Commodore 64-es a szalagra írás előtt nem keresi meg az OPEN-ben szereplő file-nevet, hanem a szalag pillanatnyi állásáról kezdi a felírást. Ez azt jelenti, hogy a file-okat

célszerű külön szalagokon tárolni, egyébként ugyanis könnyen előfordulhat, hogy akaratlanul felülírunk egy számunkra fontos adathalmazt.

A háttértárolók fejlődésének következő állomását a mágneslemez jelentette. A fejlődés egy olyan fázishoz ért, amikor a mágnesszalagos adattárolás már nem elégítette ki az igényeket, technikailag ismét új megoldásra volt szükség, és az új megoldást a mágneslemez jelentette. A Commodore 64-eshez több lemezegységet lehet kapcsolni. Az adathordozó ebben az esetben a hanglemezhez hasonló, de mágnesezett lemez, amely ósétől alapvetően abban különbözik, hogy írni is lehet rá. A lemezre írás, ill. lemezre olvasás szintaxisa:

OPEN 2, 8, 2, "0: CBM 64 FILE, S, W" ill.

OPEN 2, 8, 2, "0: CBM 64 FILE, S, R"

Mindkét utasításban az első 2-es a Commodore 64-es által használt belső file-szám, a 8-as a lemezegység száma, a második 2-es pedig az ún. csatornaszám. Az idézőjelek között álló szöveg rendre a következőket jelenti: a 0 – a meghajtószám (kettős meghajtónál 0 vagy 1), ezt követi a file neve, majd az S a file típusára utal (jelen esetben soros file-típusra), végül az R, ill. a W a READ és a WRITE rövidítése (READ olvasásnál, WRITE írásnál).

A lemezes és szalagos háttértárak közötti különbség:

– Költségek

A kazettás egység jóval olcsóbb a lemezegységnél, bár az utóbbi időben a lemezegység ára is jelentősen csökkent. Az árkülönbözetet ellensúlyozza az adattárolási kapacitás: egy lemezen 170 000 karakter tárolható, míg ugyanennyi adatot kb. négy db, C-60-as kazettán tudunk elhelyezni.

– Elérési idő

Ebben a vonatkozásban nagyságrendbeli különbség van a kétféle háttértároló között. Míg a mágnesszalagról egy 10 kbyte-os programot 200 s alatt tudunk betölteni, addig lemezzről 20 s alatt. Ugyanígyen eltérést tapasztalhatunk a soros file-ok olvasási sebességét illetően. Egy 50 címet tartalmazó soros file-t, amelyben címenként 100 betűt/jelet tárolunk, szalagon, 180, lemezen 18 s alatt tudunk végigolvasni.

– Eltérési módok, programozhatóság

Míg a szalagon a file-okat és a programokat csak sorosan tárolhatjuk, a lemezen közvetlen hozzáférésű és ún. relatív file-okat is létrehozhatunk, amelyek a soros tárolásnál jóval rugalmasabb adatkezelést biztosítanak.

Az összehasonlításból azt a következtetést vonhatjuk le, hogy a kazettás egység kisebb anyagi befektetést igényel, és a kezdők számára kiváló háttértároló eszköz. De mindazok számára, akik üzleti célból vásárolják meg a számítógépet, előbb-utóbb elengedhetetlenül fontos lesz a kényelmes, gyors adattárolás, amelyet csak a lemezegységgel lehet megoldani. A következőkben először a kazettás egységen történő adattárolás technikai elemeivel ismerkedünk meg, majd bemutatjuk a különböző elérési módokat és file-típusokat.

A kazettás egység – hogyan kerülnek fel a bitek a szalagra?

A következőkben arra szeretnénk rávilágítani, hogy az információ elemi egységeit fizikailag hogyan tároljuk a szalagon. A téma jellegéből adódik, hogy a magyarázatok során nem kerülhetjük el a műszaki vonatkozásokat.

Az információ (bit) tárolásának Commodore 64-esen alkalmazott módszerét PPM-nek nevezzük. A PPM a PULSE POSITION MODULATION rövidítése, ami azt jelenti, hogy a Commodore 64, éppúgy mint fiatalabb testvérei, a digitális jeleket nem hangfrekvenciás jelek formájában írják fel a szalagra, hanem közvetlenül.

Az információt három különböző hosszúságú (H = hosszú, R = rövid, K = közepes) digitális jel kombinációjából képzett három összetett jel ábrázolja, amelyek jelentése:

HHKK = BYTE

KKRR = 1

RRKK = 0

A jelkombinációkból az A betű pl. a következőképpen állítható elő a szalagon:

| HHKK | KKRR | RRKK | RRKK | RRKK | RRKK | RRKK | KKRR | RRKK | KKRR |
|-------|------|------|------|------|------|------|------|------|---------|
| BYTE | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| BIT # | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | paritás |

A tárolt adathalmaz szerkezete a szalagon:

Program

File

programfej

file-fej

kezdő-, végcím, név

név

programfej (ismétlés)

file-fej (ismétlés)

program (egy blokk)

adatblokk

program (ismétlés)

adatblokk (ismétlés)

utolsó blokk

utolsó blokk

A fej (header) felépítése:

Program

File

kezdőcím (XXXX)

kezdőcím (0000)

végcím (XXXX)

végcím (0000)

programnév (16 karakter)

file-név (16 karakter)

kitöltő karakterek (a pr. névben)

kitöltő karakterek (a file-névben)

Az adatblokk felépítése:

kb. 2 s üres hely

9 byte visszaszámlálás (count down) (\$89, \$88, \$87, \$86, \$85, \$84, \$83, \$82, \$81) az első adatbloknál
(\$09, \$08, \$07, \$06, \$05, \$04, \$03, \$02, \$01) ismétlésnél

Adatok

kontrollösszeg (EXOR művelet a teljes adathalmazon)

végjel (HHRR, RRRR, RRRR, RRRR, RRRR)

kb. 0.16 s üres hely

Amint azt már említettük, minden Commodore típusú gép azonos információ ábrázolási módszerrel dolgozik. Ennek ellenére a VC-20-as és a C64-es közötti adatcsere mégsem lehetséges, mivel a két géptípus eltérő órajel-frekvenciával dolgozik. A VC-20-as óráüteme nagyobb a CBM óráüteménél, a CBM üteme pedig nagyobb a Commodore 64-es üteménél. A VC-20-on, ill. a C64-esen elkészített programok nem futtathatók a nagy CBM készülékeken, de megfordítva szerencsére nem ez a helyzet, a két kis gép „megérti” a CBM gépeken tárolt programokat (file-okat). A VC-20-on megírt programokat vagy tárolt file-okat először be kell töltenünk egy CBM gépbe, majd ismét tárolnunk kell a CBM kazettás egységén. Az így kapott szalagot a VC-20-as és a C64-es egyaránt tudja értelmezni. Természetesen a programokat a géptípusnak megfelelően meg kell változtatni (pl. ki kell hagyni a C64-es vezérlőkaraktereit, módosítani kell a POKE utasítások argumentumait stb.). Sajnos eddig még nem tudtunk foglalkozni a C64-es és a VC-20-as gépek közötti programcsere szoftveres megoldásával. Ha az Olvasónak van ebben a témában hasznos ötlete, szívesen vesszük, ha azt közli velünk.

8.2 Az adatfeldolgozás alapelve: a soros file

Előzőleg áttekintettük az adathordozók fejlődéstörténetét. A következő fejezetekben a mágnesszalagos és mágneslemezes adattárolás részleteivel foglalkozunk. A két tárolási lehetőség közötti különbség jól szemlélteti majd az alapvető adatkezelési eljárások közötti különbséget is. Ebben a fejezetben a soros (szekvenciális) tárolással foglalkozunk.

A „soros” megjelölés arra utal, hogy az adatok, ill. a rekordok a file-ban egymást követően sorban vannak elhelyezve, ugyanúgy, ahogyan a zeneszámok a magnetofon szalagon. Amikor megindítjuk a hangfelvételt, a szalag attól a helytől kezdve rögzíti a zeneszámot, ahol éppen állt. Valahányszor ezt a darabot akarjuk meghallgatni, visszajátszáskor a lejátszófejet ugyanerre a pozícióra kell állítanunk.

A mágnesszalagos adattárolás alapelve nagyon hasonlít a zeneszámok tárolásának elvéhez. Sajnos minden adathalmazhoz meg kell jegyeznünk azt a szalagpozíciót, ahol a rögzítést megkezdjük, egyébként ugyanis hosszú órákba telne, amíg valamit megtalálunk a szalagon. Ennél nagyobb gondot jelent, ha nem egy file-t, hanem azon belül egy rekordot akarunk megkeresni. Az teljes képtelenség lenne, hogy a rekordok kezdetének szalagpozícióját is feljegyezzük, ennél már a lyukkártyarendszer is jobb. Egyetlen megoldás, ha a file elejétől indulva egyenként elolvassuk a rekordokat, megvizsgáljuk, hogy az éppen beolvasott rekord azonos-e a keresettel és ha nem azonos, folytatjuk az olvasást. Ha például egy 2000 rekordot tartalmazó címfile-ban egy bizonyos KOVÁCS urat keresünk, feltehetően lesz időnk néhány kávé elfogyasztására, esetleg egy kis sétára is, amíg a keresett cím megkerül. Jobban tesszük, ha a kazettás egységen viszonylag kisebb adathalmazokat tárolunk, amelyre ez a készülék kiválóan alkalmas, biztonságos és olcsó.

Aki kazettás egységen akarja az adatait tárolni, annak alkalmazkodnia kell a kazettás egység szolgáltatta lehetőségekhez. Legcélzerűbb olyan file-okat felépíteni, amelyek mérete az alapgép tárhelykapacitását nem lépi túl. Az adathalmaz kezelését ugyanis rendkívül megnehezíti és lelassítja a szalag állandó áttekerése. Ezzel szemben a tárban elhelyezett adatok közül bármelyiket pillanatok alatt meg tudjuk keresni, módosítani, majd a módosított állományt adott esetben újra tárolhatjuk a szalagon, hiszen kis adathalmaz esetén a gyakori újratárolás sem túlságosan időigényes folyamat. Nagyobb adatállományokat célszerű kisebb egységekre bontani (a címfile-t például a nevek kezdőbetűje szerint A-tól C-ig, F-től D-ig stb.), ezzel biztosítva, hogy az egyes egységek a tárban elférjenek. Könnyű belátni, hogy kazettás egységen, annak fizikai felépítése miatt, kizárólag soros szerkezetű file-ok tárolhatóak.

Természetesen soros szerkezetű file-okat a mágneslemezen is felépíthetünk, és bár az alapelve változatlan, a lemezegységen már a soros file-kezelés is bizonyos előnyöket mutat a szalagos tároláshoz képest. Ugyanis míg a szalagon egy adathalmazt a szalagpozíció ismeretében kézzel kell megkeresnünk, addig a lemezen a tartalomjegyzékben tárolt file-nevek alapján a keresést elvégzi helyettünk a gép, ráadásul gyorsabban, hiszen a szalag áttekerése sokkal több időt vesz igénybe, mint az olvasófej pozicionálása.

Hogyan programozható a soros adatfeldolgozás a Commodore 64-esen? Először meg kell nyitnunk a file-t OPEN utasítással, megadva a file-számot, egységszámot, csatornaszámot, és a file nevét. Megnyitás után megfelelő utasításokkal írhatunk a file-ba vagy olvashatunk belőle, de az lehetetlen, hogy írjunk és olvassunk egyszerre. Sőt, ha eldöntöttük, hogy egy programon belül írunk a file-ba, azt sem tehetjük meg minden további nélkül. Ha ugyanis az

OPEN 2, 8, 2, "0 : CBM 64 FILE, S, W"

utasítással megnyitjuk írásra a CBM 64 FILE nevű file-t, de ugyanilyen nevű adathalmaz már van a lemezen, a DOS a FILE EXISTS hibaüzenetet küldi, és a program futása megszakad. Ha valóban az a szándékunk, hogy a már létező adathalmazt felülírjuk, az utasítást a következőképpen kell módosítanunk:

```
OPEN 2, 8, 2, "@ 0 : CBM 64 FILE, S, W"
```

amelyben a @ jel közli a DOS-szal, hogy a már létező file-ra a továbbiakban nincs szükség. Nagyon fontos tudnunk: még a lemezen tárolt soros file-t sem tudjuk közvetlenül megváltoztatni! Először be kell olvasnunk az egész adathalmazt a tárba, ott el kell végeznünk a kívánt módosításokat, majd ezt követően az egészet a régi helyén újra tárolnunk kell.

A soros file-ok lemezes tárolása a szalagos tároláshoz képest a következő előnyökkel rendelkezik: a nagyobb tárolási és visszaolvasási sebesség, az egyszerű bővíthetőség. (Nem kell az adathalmazt a tárba beolvasni, ott kiegészíteni, majd ismét az egészet tárolni.) A bővítés utasítása:

```
OPEN 2,8,2, "0:CBM 64 FILE, S, A"
```

ahol az A az append (bővítés) angol szó rövidítése.

Az OPEN-t követő írási utasítások az adatokat a már létező file végéhez illesztik.

Ugyanez a lehetőség szalagos soros file esetében sajnos nem adott. Ugyanolyan módszerrel, ahogyan a lemezen a soros file belső adatait módosítjuk, a szalagos file-t teljes egészében a tárban kell elhelyezni, ott módosítani vagy bővíteni, majd újra tárolni.

Az eddigi elemzések alapján biztosan minden Olvasó belátta, hogy a szalagos adattárolási módszer komolyabb adatfeldolgozási feladatok megoldására valóban nem alkalmas.

Most bemutatunk néhány mintaprogramot a szalagos és lemezes soros file-ok kezelésére. A programokat bárki könnyen saját elképzelésének megfelelően átalakíthatja.

Nézzük először a **szalagos változatokat**:

1. Adatok felírása

```
0 REM **** P64. ****
1 :
2 :
10 REM *****
20 REM VEZETEKNEV ES KERESZTNEV FELIRASA A SZALAGRA
30 REM A COMMODORE-64-ES KAZETTAS EGYSEGRE KESZULT VALTOZAT
40 REM *****
50 PRINT CHR$(147): REM KEPERNYOTORLES
52 PRINT "A FILE MEGNYITASA IRASRA"
54 PRINT
56 OPEN 1,1,1,"CBM 64 FILE"
60 INPUT "VEZETEKNEV" : " ;NA$
70 INPUT "KERESZTNEV" : " ;VN$
80 PRINT
90 PRINT "FELIROM A "NA$" VEZETEKNEVET"
100 PRINT "FELIROM A "VN$" KERESZTNEVET"
110 PRINT
120 PRINT#1,NA$
130 PRINT#1,VN$
140 PRINT
150 INPUT "TOVABB (I/N) " ;JN$
155 PRINT
160 IF JN$="I" THEN GOTO 60
170 IF JN$="N" THEN GOTO 200
```

```

180 PRINT "HIBAS ADATBEVITEL !"
190 GOTO 140
200 CLOSE 1
210 END

```

A program feladata tetszőleges számú név és cím tárolása mágnesszalagon a „CBM 64 FILE” nevű file-ban. Az Olvasót feltehetően nem kell arra figyelmeztetni, hogy a program csak kazettás egységgel használható.

A következő program feladata az előzőnek pontosan az ellenkezője. Ez a program a szalagon korábban tárolt adatokat beolvassa és megjeleníti képernyőn (vagy nyomtatón). A program indításakor a szalagpozíciót a file elejére kell állítani.

2. Adatok olvasása

```

0 REM **** P65. ****
1 :
2 :
10 REM *****
20 REM A VEZETEKNEV ES KERESZTNEV VISSZAOLVASASA SZALAGROL
30 REM A COMMODORE-64-ES KAZETTAS EGYSEGRE KESZULT VALTOZAT
40 REM *****
50 PRINT CHR$(147): REM KEPERNYOTORLES
52 PRINT"A FILE MEGNYITASA OLVASASRA"
54 PRINT
56 OPEN 1,1,0,"CBM 64 FILE"
60 INPUT#1,NA$
70 INPUT#1,VN$
80 IF ST AND 64 THEN GOTO 130: REM FILE-VEGE ?
90 PRINT "A BEOLVASOTT VEZETEKNEV = ";NA$
100 PRINT "A BEOLVASOTT KERESZTNEV = ";VN$
110 PRINT
120 GOTO 60
130 PRINT "A FILE VEGEN TALALT VEZETEKNEV = ";NA$
140 PRINT "A FILE VEGEN TALALT KERESZTNEV = ";VN$
150 CLOSE 1
155 PRINT
160 END

```

Ha a fenti programmal nem képernyőre, hanem nyomtatóra szeretnénk íratni az adatokat, a következő módosításokat kell elvégezni:

```

0 REM **** P66. ****
1 :
2 :
58 OPEN 4,4
90 PRINT#4,"A BEOLVASOTT VEZETEKNEV = ";NA$
100 PRINT#4,"A BEOLVASOTT KERESZTNEV = ";VN$
110 PRINT#4
120 GOTO 60
130 PRINT#4,"A FILE VEGEN TALALT VEZETEKNEV = ";NA$
140 PRINT#4,"A FILE VEGEN TALALT KERESZTNEV = ";VN$
155 CLOSE 4

```

A harmadik program a szalagos file-t bővíti a már korábbiakban ismertetett módszerrel. A bővítés tulajdonképpen a tárban történik: az eredeti adathalmazt olvasás közben elhelyezzük két indexes változóba, és beolvasás után az első szabad indextől kezdve folytatjuk a változó feltöltését. Bővítés után a teljes adathalmazt újra tároljuk.

3. Adathalmaz bővítése

```
0 REM **** P67. ****
1 :
2 :
10 REM *****
20 REM A SZALAGOS FILE BOVITESE
30 REM A COMMODORE-64-ES KAZETTAS EGYSEGRE KESZULT VALTOZAT
40 REM *****
50 PRINT CHR$(147): REM KEPERNYOTORLES
52 PRINT "A FILE MEGNYITASA OLVASASRA"
54 PRINT
56 OPEN 1,1,0,"CBM 64 FILE"
60 DIM NA$(100),VN$(100):I=1:REM MAXIMUM 100 NEVEK 64 JOVAL 100
70 INPUT#1,NA$: NA$(I)=NA$
80 INPUT#1,VN$: VN$(I)=VN$
90 IF ST AND 64 THEN GOTO 130
100 IF I=100 THEN GOTO 130
110 I=I+1
120 GOTO 70
130 VEGE=I
140 PRINT "CSEVELJE VISSZA A SZALAGOT !"
150 PRINT
160 INPUT "KESZ (I/N) ";JN$
170 IF JN$="N" THEN GOTO 130
180 IF JN$="I" THEN GOTO 210
190 PRINT "HIBAS ADATBEVITEL !"
200 GOTO 150
210 PRINT "A FILE MEGNYITASA BOVITESRE"
220 PRINT
230 OPEN 1,1,1,"CBM 64 FILE"
240 FOR I=1 TO VEGE
250 PRINT#1,NA$(I)
260 PRINT#1,VN$(I)
270 NEXT I
280 PRINT "BOVITES"
290 PRINT
300 INPUT "VEZETEKNEV : ";NA$
310 INPUT "KERESZTNEV : ";VN$
320 PRINT
330 PRINT "FELIROM A "NA$" VEZETEKNEVET"
340 PRINT "FELIROM A "VN$" KERESZTNEVET"
350 PRINT
360 PRINT#1,NA$
370 PRINT#1,VN$
380 PRINT
390 INPUT "TOVABB (I/N) ";IN$
400 IF IN$="I" THEN GOTO 300
410 IF IN$="N" THEN GOTO 440
420 PRINT "HIBAS ADATBEVITEL !"
430 GOTO 380
440 CLOSE 1
450 END
```

Az ismertetett három program egy címnyilvántartó rendszert alkot, gyakorlatilag mindent tud, ami az adatfeldolgozáshoz szükséges: feltölti, módosítja, visszaolvassa az adathalmazt.

Most térjünk át a soros file-ok **lemezes** kezelésére.

1. Adatok felírása

Az alábbi program szinte szóról szóra azonos azzal, amit a kazettás egység kapcsán már megismertünk. Egyetlen eltérés az OPEN utasítás formátumában van.

```

0 REM **** P68. ****
1 :
2 :
10 REM *****
20 REM VEZETEKNEV ES KERESZTNEV FELIRASA LEMEZRE
30 REM A VC 1541/COMMODORE-64-RE KESZULT VALTOZAT
40 REM *****
50 PRINT CHR$(147);REM KEPERNYOTORLES
52 PRINT "A FILE MEGNYITASA IRASRA"
54 PRINT
56 OPEN 2,8,2,"CBM 64 FILE,S,W"
60 INPUT "VEZETEKNEV : ";NA$
70 INPUT "KERESZTNEV : ";VN$
80 PRINT
90 PRINT "FELIROM A "NA$" VEZETEKNEVET"
100 PRINT "FELIROM A "VN$" KERESZTNEVET"
110 PRINT
120 PRINT#2,NA$
130 PRINT#2,VN$
140 PRINT
150 INPUT "ISMETLI (I/N) ";JN$
155 PRINT
160 IF JN$="I" THEN GOTO 60
170 IF JN$="N" THEN GOTO 200
180 PRINT "HIBAS ADATBEVITEL !"
190 GOTO 140
200 CLOSE 2
210 END

```

A program folyamatosan kéri a tárolandó adatokat, mindaddig, amíg a gépkezelő kívánja. Természetesen, ha az adathalmaz mérete meghaladja a lemez kapacitását, hibaüzenetet kapunk és a program futása megszakad.

Az ilyen hibák elkerülésére célszerű adatbevitel előtt pontosan kiszámítani a szabad tárolókapacitást.

A VC-1541-es lemezegység teljes tárlkapacitása 174.848 byte (karakter). Ebből felhasználható:

Soros file tárolására: 168.656 karakter

Relatív file tárolására: 167.132 karakter

Ugyanakkor összesen 144 különböző file-t (programot) tárolhatunk a lemezen (ennyi a file-bejegyzések maximális száma a tartalomjegyzékben).

Hogyan tervezzük meg az adathalmaz méretét?

Tegyük fel, hogy a címnyilvántartásban egy rekordon belül a következő adatmezőket szeretnénk tárolni:

| Megnevezés | Hossz |
|---------------|-------|
| Sorszám | 3 |
| Vezetéknév | 20 |
| Keresztnév | 20 |
| Utca, házszám | 25 |
| Kerület | 4 |

| | |
|------------|-----|
| Város | 25 |
| Ország | 3 |
| Telefon | 16 |
| Megjegyzés | 50 |
| Összesen | 166 |

Minden adatmezőt egy RETURN karakter (CHR\$ (13)) követ, tehát a teljes rekordhossz 175 karakter. A következő képlettel meghatározhatjuk, hogy az ilyen szerkezetű rekordokból maximálisan hány fér el egy üres lemezen:

$$\text{MAX} = \text{A soros tároláskor rendelkezésre álló byte-ok száma} / \text{a rekordhossz}$$

A mi példánkban:

$$166.656 / 175 = 963,74857$$

A számítás eredménye azt mutatja, hogy 968,74857 számú rekord elfér a lemezen, de mindig célszerű egy kis ráhagyással dolgozni, tehát pl. 950 rekordra szabni a file méretét. Ha ennél jóval nagyobb adathalmazokkal kívánunk dolgozni, érdemes egy nagyobb kapacitású lemezegységet vásárolni. A 8250-es lemezegységen pl. ugyanilyen szerkezetű címrekordokból 5500-at tudunk elhelyezni.

A nagy CBM lemezegységeket az IEC-bus segítségével csatlakoztathatjuk a C64-es alapgéphez.

A DATA BECKER által forgalmazott IEC-Bus óriási előnye, hogy tartalmazza a BASIC 4.0 változatot, ami a nagy CBM gépek programozási nyelve.

2. Adatok olvasása

A lemezen tárolt soros file visszaolvasására szolgáló program ismét nem sokban különbözik a kazettás egységre vonatkozó változattól:

```

0 REM **** P69. ****
1 :
2 :
10 REM *****
20 REM VEZETEKNEV ES KERESZTNEV VISSZAOLVASASA LEMEZROL
30 REM A VC'1541/COMMODORE-64-RE KESZULT VALTOZAT
40 REM *****
50 PRINT CHR$(147):REM KEPERNYOTORLES
52 PRINT "A FILE MEGNYITASA OLVASASRA"
54 PRINT
56 OPEN 2,8,2,"CBM 64 FILE,S,R"
60 INPUT#2,NA$
70 INPUT#2,VN$
80 IF ST AND 64 THEN GOTO 130:REM FILE-VEGE ?
90 PRINT "A BEOLVASOTT VEZETEKNEV   = ";NA$
100 PRINT "A BEOLVASOTT KERESZTNEV  = ";VN$
110 PRINT
120 GOTO 60
130 PRINT "A FILE VEGEN TALALT VEZETEKNEV = ";NA$
140 PRINT "A FILE VEGEN TALALT KERESZTNEV = ";VN$
150 CLOSE 2
160 END

```

A következő programban bemutatjuk, hogy hogyan lehet a soros file-t a lemezen bővíteni.

3. Adathalmaz bővítése

```
0 REM *** P70. ***
1 :
2 :
10 REM *****
20 REM LEMEZEZ FILE BOVITESE
30 REM A VC 1541/COMMODORE-64-RE KESZULT VALTOZAT
40 REM *****
50 PRINT CHR$(147);REM KEPERNYOTORLES
52 PRINT "LEMEZ.MEGNYITASA OLVASASRA"
54 PRINT
56 OPEN 2,8,2,"CBM 64 FILE,S,A"
60 INPUT "VEZETEKNEV : ";NA$
70 INPUT "KERESZTNEV : ";VN$
80 PRINT
90 PRINT "FELIROM A "NA$" VEZETEKNEVET"
100 PRINT "FELIROM A "VN$" KERESZTNEVET"
110 PRINT
120 PRINT#2,NA$
130 PRINT#2,VN$
140 PRINT
150 INPUT "ISMETLI (I/N) ";JN$
155 PRINT
160 IF JN$="I" THEN GOTO 60
170 IF JN$="N" THEN GOTO 200
180 PRINT "HIBAS ADATBEVITEL !"
190 GOTO 140
200 CLOSE 2
210 END
```

Az Olvasónak bizonyára feltűnt, hogy a fenti, bővítő program a lemeze írás programjától mindössze egyetlen sorban, azaz az OPEN utasításban különbözik.

Lemeze írásnál az:

```
OPEN 2,8,2, "CBM 64 FILE, S, W"
```

bővítésnél pedig az:

```
OPEN 2,8,2, "CBM 64 FILE, S,A"
```

utasítást használtuk. Az utóbbit a DOS úgy értelmezi, hogy a PRINT # 2 utasításban megadott változók tartalmát nem a file elejétől írja fel a lemeze, felülírva az ott lévő adatokat, hanem a file eddigi utolsó adata mögé. Ez a bővítési lehetőség kényelmesebbé teszi a soros file karbantartásának programozását.

A fejezet utolsó részében szeretnénk a soros file néhány valóban hasznos alkalmazási területére rámutatni. Ha az adatfeldolgozási feladat olyan, hogy megköveteli az adatok gyors elérését, gyakori megváltoztatását, és ráadásul a file mérete meghaladja a tár kapacitását, a soros adattárolás nem túlságosan előnyös.

Ha olyan adatokat tárolunk, amelyeket esetleg többféle személyi számítógéppel szeretnénk feldolgoztatni, a soros file kiválóan alkalmazható, hiszen szinte minden gép ASCII kódokat használ az adatok lemezeirésznél, tehát az egyik gép el tudja olvasni a másik által felírt

információt. Ugyanezt nem mondhatjuk el a relatív, ill. közvetlen elérésű file-okról, hiszen az adatok közvetlen elérését a különböző lemezegységek operációs rendszerei általában eltérő módon szervezik meg. A soros file másik célszerű felhasználási területe a könyvelési naplók, pénzügyi nyilvántartások készítése, amelyeket folyamatosan, pl. számlák feldolgozása közben tárolunk, és amelyben az egyszer lemezre felírt adatokat nemcsak nem szükséges, hanem egyenesen tilos megváltoztatni.

8.3 Így gyorsabban megy: a relatív file-szervezés

Az adattárolás a relatív file-ban jóval kényelmesebb és gyorsabb, mint az eddigiekben megismert soros szervezésű módszer. Különösen akkor, ha olyan gépen dolgozunk, amelyen van COMMODORE 64 MAXI adatkezelő rendszer, vagy DATA BECKER IEC-Bus, amellyel a BASIC 4.0 változat utasításait használhatjuk. A relatív file-ban minden rekordhoz egy szám tartozik (vagy ún. *kulcs*), amely a file kezdetéhez viszonyítva megadja a rekord helyzetét. A kulcs alapján a file bármely rekordját közvetlenül beolvashatjuk. A közvetlen hozzáférést általában kétféleképpen szervezik a programozók:

1) A rekord valamely adatmezőjét használják kulcsként. Pl. ha egy számlanyilvántartásban a számlaszám a rekord kulcsa, így egy adott számlát a számlaszám alapján visszakereshetünk. Az anyagnyilvántartásban pl. az anyag cikkszám a tekinthető rekord-kulcsnak. A kulcs megválasztásánál két dologra kell feltétlenül figyelni: egyrészt nem szerepelhet a file-ban két azonos kulccsal ellátott rekord (ugyanis ekkor az egyik rekord elvész), másrészt a kulcsnak lehetőleg folyamatosan kell növekednie a teljes rekordhalmazon, egyébként hézagok lesz a file.

2) Ha a rekord nem tartalmaz olyan mezőt, amely betölthetné a kulcs szerepét, mert pl. minden mező fűzér, akkor általában egy ún. kulcstáblázatot készítenek. Ha pl. a címfile-ban leggyakrabban a személy neve alapján fogunk keresni, akkor célszerű a kulcstáblázatot a név mezőre felépíteni. A kulcstáblázat a file-beli sorrendben tartalmazza minden rekordból a név mezőt. A közvetlen elérés tehát úgy történhet, hogy először megkeressük, hogy a keresett név a táblázatban hányadik helyen szerepel. A kapott sorszám egyben a keresett rekord file-beli kulcsa. A kulcs alapján beolvashatjuk az egész rekordot.

A BASIC 4.0 változat, a MAXI 64 vagy a MASTER 64 szoftverek a relatív file kezelését nagyon rugalmassá, kényelmessé teszik. Annak ellenére azonban, hogy a Commodore 64-es BASIC utasításkészlete nem tartalmaz lemezkezelést, a VC 1541-es lemezegység operációs rendszerébe, a DOS-ba be van építve a relatív file-típus használatának lehetősége, a rekordok közvetlen elérésének műveletei. A CHR\$ függvény argumentumaként megadhatjuk annak a rekordnak a kulcsát, amelyre írni, vagy amelyből olvasni akarunk. Az egész folyamat két lépésből áll:

1) A relatív file megnyitása OPEN utasítással

OPEN file-szám, egységszám, csatornaszám, "név, L" + CHR\$(rekordhossz)

Az OPEN utasítás első három paramétere azonos a soros file megnyitásához szükséges paraméterekkel. A file nevét most L betű követi, erről ismeri fel a DOS, hogy relatív file megnyitása következik (az angol LENGTH szó kezdőbetűje, jelentése magyarul: hossz).

Az L betűt egy CHR\$ függvény követi, amelynek argumentuma megadja a relatív file rekordjainak hosszát. Ha például az előző fejezetben megtervezett címnilyvántartási adatokat relatív file-ban kívánjuk tárolni, rekordhosszként 175-öt kell megadnunk. A Commodore 64-es, ill. a lemezegység operációs rendszere maximum 254 karaktert tartalmazó rekordokat tud kezelni. Ha 254-nél több karaktert kell összefüggően tárolnunk, akkor vagy fel kell bontanunk a tényleges rekordot két relatív file-beli rekordra, és programból a két fizikai rekordot egy logikai rekordként kezelni, vagy a két részt eleve külön relatív file-ban kell tárolnunk.

2) Pozicionálás az adott kulcsú rekordra

PRINT # parancscsatorna száma, "P" + CHR\$ (csatornaszám) + CHR\$ (alsó) + CHR\$ (felső)

Az utasításban a "P" a POSITION angol szó kezdőbetűje, ez közli a DOS-szal, hogy az utolsó két CHR\$ függvény argumentuma a keresett rekord kulcsát (a file kezdetéhez viszonyított sorszámát) adja meg, alsó és felső byte-ra bontva.

Az utasítást még egy CHR\$ függvénnyel bővíthetnénk, argumentumában megadva, hogy a rekordon belül hányadik pozíciótól kezdve kívánunk írni vagy olvasni. Ez azt jelenti, hogy ha a rekorból pl. csak a harmadik adatmezőre van szükségünk, és tudjuk, hogy az első két mező hány karaktert foglal le, közvetlenül elérhetjük a harmadik adatmezőt. A file felépítésénél figyelembe kell vennünk még egy nagyon fontos szempontot: a relatív file-ban éppúgy, mint a soros file-ban az adatmezőket RETURN karakterrel, CHR\$(13)-mal kell elválasztanunk egymástól, egyébként ugyanis a gép az egész rekordot egy adatmezőként érzékelné. Mivel minden PRINT utasítás automatikusan egy RETURN-t helyez a kiírt adat mögé, ez a kérdés automatikusan megoldódik, ha minden adatmezőt külön PRINT # utasítással írunk fel a lemezre. Az elméleti összefüggést természetesen könnyebb megérteni, ha minden lépést egy gyakorlati példán keresztül szemlélünk. Ezért bemutatunk egy raktárkészlet-nyilvántartó programot, amelyből az Olvasó kisebb változtatásokkal tetszőleges nyilvántartási programot készíthet (pl. cím-, kazetta-nyilvántartást stb.). Térjünk vissza azonban még néhány mondat erejéig az elméletre, nézzük meg hogyan kell kiszámítani az alsó, ill. felső byte-ot.

A számítás képlete:

$$\text{A rekord sorszáma} = \text{FELSŐ} \bullet 256 + \text{ALSÓ}$$

A felső és alsó byte-ok értékét pedig a

$$\text{FELSO} = \text{INT}(\text{rekordsorszám}/256)$$

$$\text{ALSÓ} = \text{rekordsorszám} - \text{FELSŐ} \bullet 256$$

képletekkel számíthatjuk ki.

Ha például a file 78-adik rekordjára akarunk pozicionálni, az

$$\text{FB} = \text{INT}(78/256) : \text{AB} = 78 - \text{FB} \bullet 256$$

azaz FB = 0 és AB = 78

Természetesen a fenti számítás teljesen felesleges, ha a rekord sorszáma kisebb mint 256, hiszen minden ilyen esetben a felső byte értéke nulla, az alsó byte értéke pedig azonos a rekord sorszámaival.

Végül is a file 78. rekordjára a következőképpen pozicionálhatunk:

$$\text{PRINT \# parancscsatorna, "P" + CHR\$ (csatornaszám) + CHR\$ (0) + CHR\$ (78)}$$

Az általunk készített raktárnyilvántartásban az alábbi pozicionáló utasítással

$$\text{PRINT \# 15, "P" + CHR\$ (3) + CHR\$ (231) + CHR\$ (3) + CHR\$ (1)}$$

a file 999-edik rekordjának első karakterére pozicionálunk.

Az adatok felírása előtt a relatív file-t elő kell készíteni. Előkészítéshez meg kell nyitnunk a parancssortornát (15-ös), majd pozicionálás után az adott rekordra CHR\$(255) karaktert kell írni, ebből értesül ugyanis a DOS a rekord létezéséről.

A mi példánkban a 999-edik rekordra írunk CHR\$(255) karaktert.

Előkészítés után a file tetszőleges rekordjára írhatunk adatokat, figyelembe véve, hogy hány rekord számára készítettük elő a területet. Minthogy az előkészítés 999 rekordra történt, ha ennél nagyobb sorszámu rekordra íránk, a RECORD NOT PRESENT hibaüzenetet kapnánk, mivel a DOS számára ez a rekord még nem létezik.

Térjünk rá a program működésének elemzésére.

Indítás után a program megkérdezi, hogy bekapcsoltuk-e a lemezeget, majd mindaddig vár, amíg az I billentyűt le nem nyomtuk. Válasz után ismét egy kérdést tesz fel a gép: megkérdezi, hogy új lemezzel dolgozunk-e. Ebben az esetben az „új” lemez alatt nemcsak a most megvásárolt, még formátlan lemezt érti a program, hanem olyan lemezt is, amelyen még nincs adatfile. Vigyázat! Ha a válasz „igen”, a lemezt a program mindkét esetben megformálja, lehetőleg tehát ne programlemezzel dolgozzunk!

A bevezető műveletek elvégzése után megjelenik a főmenü, amely hatféle szolgáltatást kínál a használatnak. Az adatbevitel során ügyeljünk arra, hogy az adatok hossza nem haladhatja meg a DATA utasításokban (38–82) előre rögzített hosszakat.

Ezekben a sorokban rendre először az adatmező nevét, majd az adatmező hosszát rögzítettük.

Ha az adathalmazból egy már meglévő rekordot törölni akarunk, hívjuk meg a főmenüből a módosító rutint, és az árutípus első karaktereként írunk egy @ karaktert, ez a karakter jelöli ugyanis a program számára a törölt rekordot.

Minden alprogramból visszatérhetünk a főmenübe, ha adatbevitel során cikkszám helyett a VEGE szót írjuk be.

A 15. sorban az ME indexes változó a többletérték adó százalékos értékét tartalmazza. Azért adtuk meg több lehetőséget, mert a gyakorlatban ez a kulcs változik. Amikor a program a többletérték adó kulcsát kérdezi, írunk be 1-et ha 6,5%-kal; 2-t ha 13%-kal; 3-at ha 7%-kal; és 4-et ha 14%-kal kívánunk dolgozni.

Végül szeretnénk felhívni az Olvasó figyelmét a program működésének egy sajátosságára. A cikkszámot, melynek értéke 800-tól 900-ig terjedhet, a program tulajdonképpen nem árucikknek, hanem bevételnek, ill. kiadásnak tekinti. Így a használó a különböző árucikkkel mellett a magán-, ill. bankszámláit is nyilvántarthatja. A számlákat a PENZTARBLOKKOK BEVITELE alprogram olvassa be. Ha az első számlaszám nagyobb, mint 799, akkor a program árucikk-adatokat, egyébként számlaadatokat vár.

Figyeljünk arra, hogy munka közben a nyomtató és a lemezeget mindig be legyen kapcsolva! Ha az Olvasó nem rendelkezik nyomtatóval, a programot át kell írnia (mindazokban a sorokban módosítani kell, ahol PRINT #4 utasítás – nyomtatás – szerepel).

Ha a programot több kirendeltség vagy több személy adatainak nyilvántartására szeretnénk felhasználni, célszerű minden kirendeltség, ill. személy számára külön adathalmazt fenntartani, így a teljes pénzügyi helyzetről áttekintést nyerhetünk.

Reméljük, hogy mindazok, akik nem sajnálják a fáradságot az alábbi program áttekintésére, bepillantást nyernek az adatfeldolgozási munka alapjaiba, különös tekintettel a relatív file-ban tárolt adatok kezelésére. Lehet, hogy a program első látásra bonyolultnak tűnik, ennek ellenére egészen biztosak vagyunk abban, hogy bárki, kis gyakorlás után, hasonló szerkezetű programot önállóan is el tud készíteni. (Kiegészítés: lásd 186. oldal.)

```

1 REM **** P71. ****
2 :
3 :
10 CLR
15 ME(1)=1.065*ME(2)=1.13*ME(3)=1.07*ME(4)=1.14
20 FOR I=1 TO 7:READ TD*(I),TD(I):NEXT
30 DATA "1) CIKKSZAM " ,3
32 DATA "2) ARUTIPUS " ,20
34 DATA "3) DARA " ,3
36 DATA "4) EGYSEGAR EK. " ,7
38 DATA "5) OSSZ.AR EK. " ,6
40 DATA "6) EGYSEGAR VK. " ,7
42 DATA "7) OSSZ.AR VK. " ,6
50 FOR I=1 TO 3:READ TI*(I),TI(I):NEXT
56 DATA "1) CIKKSZAM " ,3
58 DATA "2) ARUTIPUS " ,20
60 DATA "3) TECS " ,1
70 FOR I=1 TO 4:READ TT*(I),TT(I):NEXT
76 DATA "1) FILIAR NR. " ,1
78 DATA "2) DATUM " ,6
80 DATA "3) SZAMLASZAM " ,6
82 DATA "4) MERLETSZAM " ,6
100 PRINT CHR$(147)
110 PRINT"*****"
120 PRINT" ADATFELDOLGOZO PROGRAM 1.0 "
130 PRINT"*****"
140 PRINT:PRINT
150 PRINT"A LEMEZEGBE VAN KAPCSOLVA (I/N)?"
160 GOTO 1 IF A$="" THEN 180
170 IF A$("<") THEN 180
180 PRINTA$
190 OPENIS,0,15,"I0" :CLOSEI5
200 PRINT"UJ LEMEZE HELYEZETT BE (I/N)?"
210 GOTO 1 IF A$="" THEN 210
220 IF A$("<") THEN 300
222 PRINTA$
230 OPENIS,0,15,"NIADATLEMEZ,AH"
240 OPENI,0,3,"O:ARTDAT,L,"+CHR$(64)
250 PRINTI5,"P"+CHR$(3)+CHR$(231)+CHR$(3)+CHR$(1)
260 PRINTI1,CHR$(255)
270 RM=INT(167132/64)
280 CLOSEI1:CLOSEI5
300 PRINTCHR$(147)
310 PRINT"*****"
320 PRINT" ADATFELDOLGOZO PROGRAM 1.0 "
330 PRINT"*****"
340 PRINT:PRINT
345 PRINT TAB(15);"FOMENU":PRINT:PRINT
350 PRINT" 1) AZ ARUCIKKEK BEVITELE." :PRINT
355 PRINT" 2) AZ ARUCIKKEK MODOSITASA." :PRINT
360 PRINT" 3) A PENZTARBLOKKOK BEVITELE." :PRINT
365 PRINT" 4) CIKKLISTA NYOMTATASA." :PRINT
370 PRINT" 5) A KIERTEKELES NYOMTATASA." :PRINT
375 PRINT" 6) A PROGRAM VEGE." :PRINT
380 PRINT"KEREM VALASSZON. (1-6) I:"
390 GOTO 1 IF A$="" THEN 390
400 A=VAL(A$):IF A(1) OR A>6 THEN 390
410 PRINTA$
420 FOR I=1 TO 1000:NEXT I
430 ON A GOTO 1000,2000,3000,4000,5000,6000
1000 OPENIS,0,15:OPENB,0,0,"O:ADATOK"
1002 GOSUB12000
1005 PRINT CHR$(147)
1010 PRINT"*****"
1020 PRINT" ADATFELDOLGOZO PROGRAM 1.0 "
1030 PRINT"*****"
1040 PRINT:PRINT

```

```

1050 PRINT TAB(8));"AZ ARUCIKKEK BEVITELE.";PRINT;PRINT
1060 FOR I=1 TO 3
1065 TE*(I)=""
1070 PRINTTI*(I);
1080 INPUTTE*(I)
1092 IF TE*(I)="VEGE" THEN 1200
1095 IF LEN(TE*(I))>TI(I) THEN 1065
1100 NEXT
1102 FOR I=4 TO 8:TE*(I)="" :NEXT
1110 RN=VAL(TE*(1))
1120 IF RN<1 OR RN>999 THEN 1065
1130 GOSUB10000
1140 GOSUB10070
1150 GOTO1005
1200 CLOSE 8:CLOSE 15
1220 GOTO3000
2000 OPEN15,8,15:OPEN8,8,8,"0:ADATOK"
2002 GOSUB12000
2005 PRINT CHR*(147)
2010 PRINT"*****";
2020 PRINT"* ADATFELDOLGOZO PROGRAM 1.0 *";
2030 PRINT"*****";
2040 PRINT;PRINT
2050 PRINT TAB(8));"ARUCIKKEK MODOSITASA.";PRINT;PRINT
2055 TE*(I)=""
2060 PRINTTI*(I);
2070 INPUTTE*(I)
2080 PRINT
2090 IF TE*(I)="VEGE" THEN 2400
2100 IF LEN(TE*(I))>TI(I) THEN 2055
2110 RN=VAL(TE*(1))
2120 IF RN<1 OR RN>999 THEN 2005
2130 GOSUB10000
2140 GOSUB10030
2142 IF VAL(TE*(1))<>RN THEN 2005
2150 PRINT CHR*(147)
2160 PRINT"*****";
2170 PRINT"* ADATFELDOLGOZO PROGRAM 1.0 *";
2180 PRINT"*****";
2190 PRINT;PRINT
2200 PRINT TAB(8));"ARUCIKKEK MODOSITASA.";PRINT;PRINT
2210 FOR I=1 TO 3
2220 PRINTTI*(I);"?";
2230 PRINTTE*(I)
2250 PRINTTI*(I);
2260 INPUTTE*(I)
2270 PRINT
2280 IF TE*(I)="VEGE" THEN 2400
2290 IF LEN(TE*(I))>TI(I) THEN 2250
2300 NEXT
2310 RN=VAL(TE*(1))
2320 IF RN<1 OR RN>999 THEN 2005
2330 GOSUB10000
2340 GOSUB10070
2400 CLOSE 4:CLOSE 8:CLOSE 15
2430 GOTO3000
2530 GOTO3005
3000 GOSUB8000:OPEN15,8,15:OPEN8,8,8,"0:ADATOK"
3002 GOSUB12000
3005 PRINT CHR*(147)
3010 PRINT"*****";
3020 PRINT"* ADATFELDOLGOZO PROGRAM 1.0 *";
3030 PRINT"*****";
3040 PRINT;PRINT
3050 PRINT TAB(8));"A PENZTARBLOKKOK BEVITELE.";PRINT;PRINT
3060 TE*(I)=""
3070 PRINTTI*(I);

```

```

3080 INPUTTE*(1)
3090 PRINT
3100 IF TE*(1)="VEGE" THEN 3700
3110 IF LEN(TE*(1))>TI(1) THEN 3060
3120 RN=VAL(TE*(1))
3130 IF RN<1 OR RN>899 THEN 3005
3132 IF DW=1 AND RN>799 THEN 3005
3134 IF DW=2 AND RN<800 THEN 3005
3140 GOSUB10000
3150 GOSUB10030
3152 IF VAL(TE*(1))<RN THEN 3005
3154 IF LEFT*(TE*(2),1)="Q" THEN 3005
3160 PRINT CHR*(147)
3170 PRINT"*****";
3180 PRINT"      ADATFELDOLGOZO PROGRAM 1.0      ";
3190 PRINT"*****";
3200 PRINT:PRINT
3210 PRINT TAB(8);"A PENZTARBLOKKOK BEVITELE.";PRINT:PRINT
3212 FOR I=1 TO 5
3214 TH*(1)=TE*(I+3)
3216 NEXT
3220 FOR I=1 TO 2
3230 PRINTTD*(I);"?";TE*(I)
3235 TX*(1)=TE*(I)
3240 PRINT
3250 NEXT
3255 TX*(3)=TE*(3)
3260 PRINTTD*(3);
3270 INPUTTX*(4)
3275 TE*(4)=TX*(4)
3280 PRINT
3285 IF VAL(TE*(4))<-899 OR VAL(TE*(4))>999 THEN 3260
3287 IF LEN(TE*(4))>TD(3) THEN 3260
3290 PRINTTD*(4);
3295 TE*(5)=""
3300 INPUTTX*(5)
3305 TE*(5)=TX*(5)
3310 PRINT
3315 IF LEN(TE*(5))>TD(4) THEN 3290
3320 PRINTTD*(6);
3325 TE*(7)=""
3330 INPUTTX*(7)
3335 TE*(7)=TX*(7)
3340 PRINT
3345 IF LEN(TE*(7))>TD(6) THEN 3320
3350 TH=VAL(TE*(5))*VAL(TE*(4))
3351 TX*(6)=STR*(TH)
3352 TH=TH+VAL(TH*(3))
3355 TE*(6)=STR*(TH)
3360 TH=VAL(TE*(7))*VAL(TE*(4))
3361 TX*(8)=STR*(TH)
3362 TH=TH+VAL(TH*(5))
3365 TE*(8)=STR*(TH)
3370 TH=VAL(TE*(5))
3371 IF VAL(TE*(4))<1 THEN TH=-TH
3372 TH=TH+VAL(TH*(2))
3375 TE*(5)=STR*(TH)
3380 TH=VAL(TE*(7))
3381 IF VAL(TE*(4))<1 THEN TH=-TH
3382 TH=TH+VAL(TH*(4))
3385 TE*(7)=STR*(TH)
3390 TH=VAL(TE*(4))
3392 TH=TH+VAL(TH*(1))
3395 TE*(4)=STR*(TH)
3400 RN=VAL(TE*(1))
3470 GOSUB 10000
3480 GOSUB 10070

```

```

3485 FOR I=1 TO 8:TE*(I)=TX*(I):TX*(I)="":NEXT
3490 IF DW=0 AND VAL(TE*(1))<800 THEN DW=1:GOSUB5360:GOTO3510
3500 IF DW=0 AND VAL(TE*(1))>799 THEN DW=2:GOSUB7005:GOTO3520
3510 IF DW=1 THEN GOSUB5320:GOTO3530
3520 IF DW=2 THEN GOSUB7120
3530 GOTO3005
3700 IF DW=1 THEN GOSUB5500:GOTO3800
3710 IF DW=2 THEN GOSUB7190
3800 DW=0:DV=0
3899 CLOSE 4:CLOSE 8:CLOSE 15:GOTO3000
4000 OPEN15,8,15:OPEN0,8,0,"D:ADATOK"
4002 GOSUB12000
4005 PRINT CHR$(147)
4010 PRINT"*****"
4020 PRINT"*      ADATFELDOLGOZO PROGRAM 1.0      *"
4030 PRINT"*****"
4040 PRINT:PRINT
4050 PRINT TAB(8);"ARUCIKK-LISTA NYOMTATASA.":PRINT:PRINT
4060 FOR I=1 TO 2
4070 TE*(I)=" "
4080 PRINTTT*(I);
4090 INPUTTE*(I)
4100 PRINT
4110 IF TE*(I)="VEGE" THEN 4999
4120 IF LEN(TE*(I))>TT(I) THEN 4070
4130 NEXT
4135 TE*(3)="":TE*(4)=" "
4140 IF DW=1 THEN RETURN
4200 PRINT CHR$(147)
4210 PRINT"*****"
4220 PRINT"*      ADATFELDOLGOZO PROGRAM 1.0      *"
4230 PRINT"*****"
4240 PRINT:PRINT
4250 PRINT TAB(8);"AZ ARUCIKK-LISTA NYOMTATASA.":PRINT:PRINT
4260 PRINT"A NYOMTATO BE VAN KAPCSOLVA (I/N)? ";
4270 GETA$:IF A$="" THEN 4270
4280 IF A$<"I" THEN 4270
4290 PRINTA$
4300 OPEN4,4
4310 PRINT#4,"FILIALE NR. ";TE*(1);
4320 PRINT#4,CHR$(16);"20";"DATUM. ";TE*(2);
4340 PRINT#4,CHR$(16);"40";"SZAMLA SZAM ";TE*(3);
4345 PRINT#4,CHR$(16);"60";"TECS";TE*(4)
4350 PRINT#4,
4360 PRINT#4,"CIKK SZAM ";
4375 PRINT#4,CHR$(16);"20";"ARUTIPUS";
4385 PRINT#4,CHR$(16);"60";"TECS"
4390 PRINT#4,"-----";
4405 PRINT#4,CHR$(16);"20";"-----";
4415 PRINT#4,CHR$(16);"60";"-----"
4420 FOR RN=1 TO 999
4430 GOSUB 10000
4440 GOSUB 10030
4442 IF VAL(TE*(1))<>RN THEN 4480
4444 IF LEFT$(TE*(2),1)="0" THEN 4480
4450 PRINT#4,TE*(1);
4460 PRINT#4,CHR$(16);"20";TE*(2);
4470 PRINT#4,CHR$(16);"60";TE*(3)
4480 NEXT
4490 FOR I=1 TO 3
4492 PRINT#4,
4494 NEXT
4999 CLOSE 4:CLOSE 8:CLOSE 15:GOTO3000
5000 OPEN15,8,15:OPEN0,8,0,"D:ADATOK"
5002 GOSUB12000
5005 PRINT CHR$(147)
5010 PRINT"*****"

```

```

5020 PRINT"* ADATFELDOLGOZO PROGRAM 1.0 *";
5030 PRINT"*****";
5040 PRINT:PRINT
5050 PRINT TAB(8);"A KIERTEKELES NYOMTATORA.";PRINT:PRINT
5060 FOR I=1 TO 4
5070 TE*(I)=""
5080 PRINTTT*(I);
5090 INPUTTE*(I)
5100 PRINT
5110 IF TE*(I)="VEGE" THEN 5080
5120 IF LEN(TE*(I))>TT*(I) THEN 5070
5130 NEXT
5140 IF DW=1 THEN RETURN
5200 PRINT CHR*(147)
5210 PRINT"*****";
5220 PRINT"* ADATFELDOLGOZO PROGRAM 1.0 *";
5230 PRINT"*****";
5240 PRINT:PRINT
5250 PRINT TAB(8);"A KIERTEKELES NYOMTATASA.";PRINT:PRINT
5252 SA=0:SE=0:SG=0:SF=0:SH=0
5255 FOR I=1 TO 4:M1(I)=0:M2(I)=0:NEXT
5260 PRINT"A NYOMTATO BE VAN KAPCSOLVA (I/N)";
5270 GETA*:IF A*="" THEN 5270
5280 IF A*<"I" THEN 5270
5290 PRINTA*
5300 OPEN#4
5310 PRINT#4,"FILIAI NR.";TE*(1);
5320 PRINT#4,CHR*(16);"20";"DATUM ";TE*(2);
5335 IF DW=0 THEN PRINT#4,100T05350
5340 PRINT#4,CHR*(16);"40";"SZAMLASZAM ";TE*(3);
5345 PRINT#4,CHR*(16);"60";"BERLETSZAM ";TE*(4)
5350 PRINT#4,
5355 IF DW=1 THEN RETURN
5360 PRINT#4,"CIK.SZ. ";
5370 PRINT#4,CHR*(16);"10";"DB. ";
5375 PRINT#4,CHR*(16);"15";"ARUTIPUS";
5380 PRINT#4,CHR*(16);"40";"EGYS.AR ";
5390 PRINT#4,CHR*(16);"50";"OSSZ.AR ";
5400 PRINT#4,CHR*(16);"60";"EGYS.AR ";
5410 PRINT#4,CHR*(16);"70";"OSSZ.AR ";
5411 PRINT#4,CHR*(16);"42";"EK. ";
5412 PRINT#4,CHR*(16);"52";"EK. ";
5413 PRINT#4,CHR*(16);"62";"VK. ";
5414 PRINT#4,CHR*(16);"72";"VK. ";
5420 PRINT#4,"-----";
5425 PRINT#4,CHR*(16);"10";"-----";
5430 PRINT#4,CHR*(16);"15";"-----";
5440 PRINT#4,CHR*(16);"40";"-----";
5450 PRINT#4,CHR*(16);"50";"-----";
5460 PRINT#4,CHR*(16);"60";"-----";
5470 PRINT#4,CHR*(16);"70";"-----";
5475 IF DW=1 THEN RETURN
5480 FOR RN=1 TO 799
5490 GOSUB 10000
5500 GOSUB 10030
5510 IF VAL(TE*(1))<>RN THEN 5500
5515 IF LEFT*(TE*(2),1)="" THEN 5500
5520 PRINT#4,TE*(1);
5525 PRINT#4,CHR*(16);"10";TE*(4);
5530 PRINT#4,CHR*(16);"15";TE*(2);
5540 PRINT#4,CHR*(16);"40";TE*(5);
5550 PRINT#4,CHR*(16);"50";TE*(8);
5560 PRINT#4,CHR*(16);"60";TE*(7);
5570 PRINT#4,CHR*(16);"70";TE*(6)
5571 SA=SA+VAL(TE*(4))
5572 SE=SE+VAL(TE*(5))
5573 SG=SG+VAL(TE*(6))

```

```

5574 SF=SF+VAL(TE*(7))
5575 SH=SH+VAL(TE*(8))
5576 IF VAL(TE*(3))<1 OR VAL(TE*(3))>4 THEN 5579
5577 M1(VAL(TE*(3)))=M1(VAL(TE*(3)))+(VAL(TE*(8))-ME(VAL(TE*(3)))-VAL(TE*(8)))
5578 M2(VAL(TE*(3)))=M2(VAL(TE*(3)))+(VAL(TE*(8))-VAL(TE*(8))/ME(VAL(TE*(3))))
5579 IF DW=1 THEN RETURN
5580 NEXT
5581 PRINT#4,"-----";
5600 PRINT#4,CHR$(16);"10";"-----";
5605 PRINT#4,CHR$(16);"15";"-----";
5610 PRINT#4,CHR$(16);"40";"-----";
5620 PRINT#4,CHR$(16);"50";"-----";
5630 PRINT#4,CHR$(16);"60";"-----";
5640 PRINT#4,CHR$(16);"70";"-----";
5650 PRINT#4,"OSSZESEN1";
5660 PRINT#4,CHR$(16);"68";SA;
5670 PRINT#4,CHR$(16);"38";SE;
5680 PRINT#4,CHR$(16);"49";S0;
5690 PRINT#4,CHR$(16);"59";SF;
5700 PRINT#4,CHR$(16);"69";SH;
5710 FOR I=1 TO 3:PRINT#4,I;NEXT
5720 PRINT#4,"TECS. EK. 8,5 % = ";M1(1);
5725 PRINT#4,CHR$(16);"40";"TECS. EK. 7 % = ";M1(3)
5730 PRINT#4,"TECS. EK. 13,0 % = ";M1(2);
5735 PRINT#4,CHR$(16);"40";"TECS. EK. 14 % = ";M1(4)
5740 PRINT#4,"TECS. VK. 8,5 % = ";M2(1);
5745 PRINT#4,CHR$(16);"40";"TECS. VK. 7 % = ";M2(3)
5750 PRINT#4,"TECS. VK. 13,0 % = ";M2(2);
5755 PRINT#4,CHR$(16);"40";"TECS. VK. 14 % = ";M2(4)
5756 FOR I=1 TO 3:PRINT#4,I;NEXT
5760 PRINT#4,"OSSZ.AR EK. = ";S0+M1(1)+M1(2)+M1(3)+M1(4)
5770 PRINT#4,"OSSZ.AR VK. = ";SH-M2(1)-M2(2)-M2(3)-M2(4)
5775 FOR I=1 TO 3:PRINT#4,I;NEXT
5777 IF DW=1 THEN RETURN
5780 GOSUB 7000
5999 CLOSE 4:CLOSE 0:CLOSE 15:GOTO 300
6000 CLOSE 4:CLOSE 8:CLOSE 15
6030 PRINT CHR$(147)
6040 END
7000 S1=0:S2=0
7005 PRINT#4,"CIKKSZAM";
7010 PRINT#4,CHR$(16);"15";"CIM ";
7020 PRINT#4,CHR$(16);"40";"KIADAS";
7030 PRINT#4,CHR$(16);"60";"BEVETEL"
7040 PRINT#4,"-----";
7050 PRINT#4,CHR$(16);"15";"---";
7060 PRINT#4,CHR$(16);"40";"-----";
7070 PRINT#4,CHR$(16);"60";"-----";
7075 IF DW=2 THEN RETURN
7080 FOR RN=000 TO 999
7090 GOSUB 10000
7100 GOSUB 10030
7110 IF VAL(TE*(1))<>RN THEN 7180
7120 PRINT#4,TE*(1);
7130 PRINT#4,CHR$(16);"15";TE*(2);
7140 PRINT#4,CHR$(16);"40";TE*(5);
7150 PRINT#4,CHR$(16);"60";TE*(7)
7160 S1=S1+VAL(TE*(5))
7170 S2=S2+VAL(TE*(7))
7171 IF VAL(TE*(3))<1 OR VAL(TE*(3))>4 THEN 7175
7172 M2(VAL(TE*(3)))=M2(VAL(TE*(3)))+(VAL(TE*(7))-VAL(TE*(7))/ME(VAL(TE*(3))))
7173 M1(VAL(TE*(3)))=M1(VAL(TE*(3)))+(VAL(TE*(5))-VAL(TE*(5))/ME(VAL(TE*(3))))
7175 IF DW=2 THEN RETURN
7180 NEXT
7190 PRINT#4,"-----";
7200 PRINT#4,CHR$(16);"15";"---";
7210 PRINT#4,CHR$(16);"40";"-----";

```



```

7220 PRINT#4,CHR$(16);"60";"-----"
7230 PRINT#4,"OSSZESEN:";
7240 PRINT#4,CHR$(16);"38";S1;
7250 PRINT#4,CHR$(16);"59";S2
7260 FOR I=1 TO 3:PRINT#4,I;NEXT
7270 PRINT#4,"TECS. EK. 6,5 % = "J;M1(1);
7280 PRINT#4,CHR$(16);"40";"TECS. EK. 7 % = "J;M1(3)
7290 PRINT#4,"TECS. EK. 13,0 % = "J;M1(2);
7300 PRINT#4,CHR$(16);"40";"TECS. EK. 14 % = "J;M1(4)
7310 PRINT#4,"TECS. VK. 6,5 % = "J;M2(1);
7320 PRINT#4,CHR$(16);"40";"TECS. VK. 7 % = "J;M2(3)
7330 PRINT#4,"TECS. VK. 13,0 % = "J;M2(2);
7340 PRINT#4,CHR$(16);"40";"TECS. VK. 14 % = "J;M2(4)
7350 FOR I=1 TO 3:PRINT#4,I;NEXT
7360 PRINT#4,"KIADAS      = "J;S1-M1(1)-M1(2)-M1(3)-M1(4)
7370 PRINT#4,"BEVETEL    = "J;S2-M2(1)-M2(2)-M2(3)-M2(4)
7380 FOR I=1 TO 3:PRINT#4,I;NEXT
7999 RETURN
8000 PRINT CHR$(147)
8010 PRINT"*****"
8020 PRINT"*          ADATFELDOLGOZO PROGRAM 1.0          *"
8030 PRINT"*****"
8040 PRINT:PRINT
8050 PRINT TAB(8);"A PENZTARBLOKKOK BEVITELE.";PRINT:PRINT
8060 SA=0:SE=0:SF=0:SH=0:S1=0:S2=0
8070 FOR I=1 TO 4
8072 M1(I)=0
8074 M2(I)=0
8076 NEXT
8080 PRINT"A NYOMTATO BE VAN KAPCSOLVA (I/N)? ";
8090 GETA:IF A$="" THEN 8090
8100 IF A$<"I" THEN 8090
8110 PRINTA$
8120 OPEN#4
8130 PRINT CHR$(147)
8140 PRINT"*****"
8150 PRINT"*          ADATFELDOLGOZO PROGRAM 1.0          *"
8160 PRINT"*****"
8170 PRINT:PRINT
8180 PRINT TAB(8);"A PENZTARBLOKKOK BEOLVASASA.";PRINT:PRINT
8190 DW=1:GOSUB 5000
8190 DW=1:GOSUB 5310:DW=0
8190 RETURN
10000 HB=INT(RN/256):LB=RN-HB*256
10010 PRINT#15,"P"+CHR$(8)+CHR$(LB)+CHR$(HB)+CHR$(1)
10015 GOSUB12000
10020 RETURN
10030 INPUT#8,TE$(1),TE$(2),TE$(3),TE$(4),TE$(5),TE$(6),TE$(7),TE$(8)
10040 RETURN
10070 TE$=TE$(1)+CHR$(13)+TE$(2)+CHR$(13)+TE$(3)+CHR$(13)+TE$(4)+CHR$(13)
10072 TE$=TE$+TE$(5)+CHR$(13)+TE$(6)+CHR$(13)+TE$(7)+CHR$(13)+TE$(8)
10080 PRINT#8,TE$
10110 RETURN
12000 INPUT#15,X,X$,Y$,Z$
12030 IF X=0 THEN CLOSE 8:CLOSE 15:RETURN
12040 FOR I=1 TO 6000:NEXT
12060 GOTO 100

```

8.4 Egy másik módszer: a közvetlen hozzáférés

Az adatkezelésnek ezt a módját sajnos gyakran teljesen elhanyagolják, és az elhanyagolás oka abban rejlik, hogy ez a módszer valamivel bonyolultabb, mint a többi. Vajon érdemes-e mélyebben megismerni a közvetlen hozzáférési műveleteket, tulajdonképpen mire használhatók?

1) Saját szervezésű file-ok létrehozására – random file-ok

A random file-típus egyesíti a soros és relatív típusok előnyeit, azok hátrányai nélkül.

2) A lemez egyes szektorainak közvetlen kezelésére (írására/olvasására)

Az utóbbi olyan előnyökkel jár, amelynek jelentőségét az Olvasó az eddig közölt ismeretek alapján nem mérheti fel, ezért erre részletesebben kitérünk.

1. Közvetlen előérésű (random) file-ok

A soros és relatív file-okkal szemben a random file minden rekordja 256 byte – azaz egy adatblokk – hosszúságú.

A random file tehát összesen 664 blokkot tartalmazhat. Ennek ellenére kisebb rekordokkal is dolgozhatunk, ha egy blokkba több rekordot helyezünk el (négy, egyenként 64 byte-os rekordot). Adatkezelés közben arra azonban ügyelnünk kell, hogy minden műveletnél (írás/olvasás) a közvetlen hozzáférésű utasítások egy blokkot (256 byte-ot) kezelnek egy egységként. A random file szerkezetét a programozónak kell meghatároznia. Ahhoz, hogy az adatblokkokat közvetlenül feldolgozhassuk, feldolgozás előtt létre kell hoznunk egy soros file-t, amelyben az adatblokkok lemezcímeit tároljuk. Összesen három file-t kell megnyitni.

1. Az adatblokkok címét tartalmazó file-t.

2. A parancsfile-t.

3. Az adatfile-t.

```
0 REM **** P72. ****
1 :
2 :
10 OPEN 4,8,4,"CBM 64 FILE,S,W": REM SZKVENCIÁLIS ADATFILE
20 OPEN 15,8,15: REM PARANCSCSATORNA
30 OPEN 5,8,5,"#": REM ADAT-FILE
40 TE$="DATA BECKER GMBH"
50 PRINT#5,TE$;" ";1: REM SZÖVEG, REKORD
60 T=1:S=1: REM SAV=1:SZEKTOR=1
70 PRINT#15,"B-A: ";0,T,S: REM EGYSEG, SAV, SZEKTOR
80 INPUT#15,ER,NA$,TR,BL: REM A HIBACSATORNA LEOLVASASA
90 IF ER=65 THEN T=TR:S=BL:GOTO 70
100 PRINT#15,"B-W: ";5,0,T,S: REM EGY REKORD FELIRASA
110 PRINT#4,T$;" ";S
120 CLOSE 5
130 CLOSE 15
140 CLOSE 4
150 END
```

Mit csinál a fenti program? Megnyitja a szükséges három file-t, ezután elhelyez egy szöveget a TE\$ változóban, és tárolja az átmeneti adatterületen (pufferben). Az 1-es sáv 1-es szektorától indulva szabad blokkot keres a lemezen (70-es sor). A B-A utasítással megpróbálja felfoglalni az adott blokkot, majd a következő INPUT# utasítással leolvasa a hibacsatornát.

Ha a hibakód 65, akkor a blokk foglalt, tehát a keresés folytatódik. Ha a blokk szabad, a program felírja erre a területre a szöveget, majd az adatblokk címét (sávot és szektort) felírja a soros file-ba, ezzel biztosítva, hogy a továbbiakban a közvetlen hozzáférésű utasítással felírt információt visszaolvashassuk. A file-ok lezárása után a program futása véget ér.

Hogyan olvassuk vissza a felírt adatblokkot?

```
0 REM **** F73. ****
1 :
2 :
10 OPEN 4,8,4,"CBM 64 FILE"
20 OPEN 15,8,15
30 OPEN 5,8,5,"B-R"
40 INPUT#4,T,S: REM A CIM BEOLVASASA
50 PRINT#15,"B-R: ";5,0,T,S
60 INPUT#5,TE$,RE
70 PRINT#15,"B-F: ";0,T,S
80 CLOSE 5
90 CLOSE 4
100 PRINT#15,"S:CBM 64 FILE"
110 CLOSE 15
```

A file-ok megnyitása után a soros file-ból beolvassuk az adatblokk címét, majd B-R utasítással beolvassuk a szöveget a kapott lemezcímről a pufferbe, onnan a TE\$ változóba. Végül felszabadítjuk az adatblokkot – amit természetesen csak akkor kell megtenni, ha az adatblokk tartalmára a továbbiakban már nincs szükségünk. Végül lezárjuk a file-okat, töröljük a soros file-t és lezárjuk a parancscsatornát.

2. Közvetlen lemezkezelés

Ahogy a cím is mutatja, ezekkel az utasításokkal a lemez tetszőleges sávjának, szektorának tartalmát közvetlenül beolvashatjuk, ill. tetszőleges információt írhatunk bele, anélkül, hogy bármilyen file-t meg kellene nyitnunk. Beolvashatjuk pl. a lemez tartalomjegyzékét, amit eddig csak a LOAD "\$", 8 utasítással tudtunk megtenni, felülírva a tár aktuális tartalmát. Megváltoztathatjuk a lemezen tárolt programot anélkül, hogy be kellene olvasnunk az egészet a tárba – akaratlanul elrontott programokat, file-okat helyreállíthatunk stb.

Ezek a műveletek veszélyesek lehetnek azoknak, akik nem ismerik tökéletesen a lemez felépítését, a tartalomjegyzékét, a BAM-ot, ill. a programok tárolási módját. Egy rossz helyre célzott lemezeírással az egész lemez tartalmát meg lehet semmisíteni. Hogy ez ne következessen be, javasoljuk, tanulmányozza át az Olvasó „A VC 1541-es lemezegység programozása” c. könyvet.* Ha mégis kísérletezni szeretne a közvetlen lemezműveletekkel, feltétlenül olyan lemezt használjon, amelyen nem tárol értékes információt! A közvetlen hozzáférésű (random) utasítások:

Megnevezés

BLOCK-READ (1 blokk olvasás,
BLOCK-WRITE (2 blokk írás,
BLOCK-ALLOCATE (3 blokk lefoglalása),
BLOCK-FREE (4 blokk felszabadítása)
BUFFER-POINTER (5 blokkmutató beállítás)

Szintaktika

"B-R.": csatorna; egység; sáv; szektor
"B-W.": csatorna; egység; sáv; szektor
"B-A.": egység; sáv; szektor
"B-F.": egység; sáv; szektor
"B-P.": csatorna; pozíció

* Eredeti címe: Das grosse Floppy Buch (DATA BECKER, 1983). Megjelent magyarul a NOVOTRADE RT. kiadásában, 1985-ben. (A szerk.)

8.5. Egy lezáratlan file megmentése

Hozzátehetjük, hogy erre a műveletre túl gyakran nincs szükség. De ha mégis, igen nagy bosszúságot okoz. Miről van szó tulajdonképpen?

Tegyük fel, hogy Ön fáradtságos munkával megszerezte a lemezgyűjteményét (most kivételesen zeneműveket tartalmazó lemezekről van szó) és a könnyebb keresés érdekében a lemezek címét egy soros file-ban, mágneslemezen szeretné tárolni. Mi sem egyszerűbb ennél, a már megszokott módon elkészíti az adatfelvivő programot (a lemezek számától függően egy vagy több soros file-t tervezve), majd bebillentyűzi az adatokat, ami, tegyük fel, hogy ez esetben 500 lemezcímet jelent. Amikor már majdnem készen van, a kedves felesége (férje) kikapcsolja a hálózati biztosítékot. Nem tesz semmit, gondolja Ön, hiszen a programon csak apró változtatást kell tenni (a W betűt az OPEN utasításban A betűre módosítani) és máris folytathatja a munkát.

Sajnos a program indításakor hibaüzenetet küld a gép! A hibacsatorna lekérdezésére a válasz: WRITE FILE OPEN. Ha most megvizsgálja a lemez tartalomjegyzékét, látja, hogy a soros file típusmegjelölése után egy csillag helyezkedik el. Ez annyit jelent, hogy az írásra megnyitott file-t nem zárta le CLOSE utasítással. Valóban ez történt, hiszen az adatbevitel áramkimaradás miatt megszakadt, a program nem szabályosan ért véget. A hiba következménye, hogy a soros file-t semmilyen szabályos BASIC utasítással nem lehet visszaolvasni. Az Ön lemeznylévtartása tehát örökre elveszett!

A helyzet azonban mégsem ennyire reménytelen!

Bemutatunk egy olyan programot, amellyel az ilyen látszólag véglegesen megsemmisült file-okat helyre lehet állítani.

Természetesen mint mindig, most is részletesen ismertetjük a programban használt változók jelentését, és az egyes programlépéseket.

A felhasznált változók:

| | |
|------|--|
| E | A file-név pozíciója a tartalomjegyzéken belül |
| S | A random utasításhoz szükséges szektorszám |
| T | A random utasításhoz szükséges sávszám |
| TY | A file típusa (T\$-ből) |
| X | Futó változó a file-név meghatározásához |
| A\$ | Munkaváltozó az S\$ felépítéséhez |
| F\$ | A kívánt file-név |
| S\$ | A teljes szektor tartalma |
| T\$ | A file típusa |
| X\$ | A tartalomjegyzékből beolvasott, üres karakterekkel 16 karakter hosszúságúra kiegészített file-név |
| X1\$ | ua. mint X\$ |

A program lépései:

| | |
|-----|---|
| 70 | Az adatcsatorna megnyitása közvetlen eléréshez |
| 80 | A parancscsatorna megnyitása |
| 100 | A keresett file-név beolvasása |
| 110 | A sáv- és szektorszám meghatározása |
| | A VC 1541-es lemezegység a tartalomjegyzéket a lemez 18-as sávján az 1-es szektorra helyezi a CBM 4040-hez hasonlóan. A 8050-es lemezegység ezzel szemben a 39-es sávra, tehát ha az utóbbival rendelkezünk, ezt a programsort módosítani kell. |

- 120 A T, ill. S változóban megadott sáv-, ill. sektorszámú blokkot beolvassuk a pufferbe (a 0-ás egységről).
- 150 A puffer tartalmát az A\$-on keresztül (karakterenként) az S\$-ba gyűjtjük.
- 160 Egy szektor nyolc file-bejegyzést tartalmaz. Ezek között megkeressük a mi file-unkhoz tartozó bejegyzést.
- 170 A bejegyzésből leválasztjuk a file nevét (16 byte-ot), majd az X\$-ba töltjük.
- 200-210 A file-név végét a CHR\$(160) karakter jelzi (SHIFT + SPACE).
- 220-230 Ha a keresett file bejegyzését nem találtuk meg, a keresés folytatódik, ha megtaláltuk, elemezzük a bejegyzést.
- 240-260 Minden szektor elején megtalálható a tartalomjegyzék következő adatblokkjának címe (sáv, szektor), ha még ilyen van. A sáv nulla értéke jelzi, hogy nincs több folytatóblokk, a tartalomjegyzéknek vége.
- 300-310 Leválasztjuk a file-típust, majd a megfelelő numerikus értéket a T változóba töltjük.
- 320 Ha T=0, az adott bejegyzés üres.
- 360 Az itt beállított bit volt minden baj okozója. Ez a bit jelzi ugyanis, hogy egy írásra megnyitott file le van zárva, vagy nincs. (A csillagot a DOS ennek a bitnek az értéke alapján írja ki!)
- 370 Összeállítjuk a teljes szektort, amelyben a file-t lezártnak minősítettük.
- 390-410 A puffermutatót visszaállítjuk, a szektort a pufferbe, majd onnan a lemezeire írjuk.
- 420-490 Néhány sor emlékeztetőül a program feladatáról.

A program futtatása rendkívül egyszerű.

Betöltjük a tárbá, majd behelyezzük az elrontott file-t tartalmazó lemezt az egységbe (kettős meghajtó esetén a 0-ás egységbe). Elindítjuk a programot, majd beírjuk az elrontott file nevét. Ha a program megtalálja az adott nevű file-t a lemezen, a többi már az ő dolga.

Néhány korlátozás: Ezt az eljárást sajnos relatív file esetében nem használhatjuk, hiszen a file-struktúra teljesen más. A relatív file megmentése bonyolultabb program megírását és a lemezszerkezet beható ismeretét igényli.

Ha az elrontott file-t az alábbi programmal helyreállítottuk, a benne levő adatokat sorban végigolvasva átmásolhatjuk egy másik file-ba.

A másolást mindenképpen célszerű megtenni, mert bár az adatok nem vesztek el, lehet hogy a tartalomjegyzékben egyéb hiba is keletkezett a hibás programfutás közben. Másolás után teljesen nyugodtan alhat az Olvasó, a lemeznyilvántartása biztonságban van.

Reméljük, hogy ez a program nemcsak egyszeri segítséget, hanem sok tanulságot is nyújtott.

```
0 REM **** P74. ****
1 :
2 :
10 PRINT CHR$(147);
20 PRINT CHR$(5);
70 OPEN 2,8,2,"*": REM KOZVETLEN ELERES
80 OPEN 15,8,15: REM PARANCSCSATORNA
90 PRINT:PRINT
100 INPUT "FILE NEVE ";F$:PRINT:PRINT
110 T=18:S=1:REM 1541 TARTALOMJEGYZEK **T=39 A CBMB050-EN
120 PRINT#15,"U1 2 0" T;S: REM OLVASAS
130 S$="": REM A BEOLVASOTT SEKTOR
140 : REM SEKTOR
150 FOR I=1 TO 255:GET#2,A$:S$=S$+LEFT$(A$+CHR$(0),1):NEXT
160 FOR I=0 TO 7: REM 8 BEJEGYZES
170 X$=MID$(S$,I*32+6,16):X1$=X$
180 REM A FILE-NEV LEVALASZTASA
```

```

190 X=1
200 IF MID$(X$,X,1)<>CHR$(160) THEN X=X+1: IF X<17 THEN 200
210 X%=LEFT$(X$,X-1)
220 IF X%=F$ THEN E=I:GOTO 300
230 NEXT I
240 T=ASC(S$):S=ASC(MID$(S$,2,1))
250 REM A KOVETKEZO SZEKTOR OLVASASA
260 IF T<>0 THEN 120
270 REM VEGE
280 PRINT "AZ "F$" FILE NINCS EZEN A LEMEZEN !"
290 CLOSE 2:CLOSE 15:END
300 T%=MID$(S$,E*32+3)
310 TY=ASC(T$) AND 15
320 IF TY=0 THEN NEXT I:GOTO 240
330 IF TY<>4 THEN 340
335 PRINT "RELATIV FILE-T NEM TUDOK MASOLNI !"
337 GOTO 290
340 TY$="DELSEQPRGUSRREL"
350 PRINT "FILE "X1$" "MID$(TY$,TY*3+1,3):PRINT
360 T$=CHR$(ASC(T$) OR 128)
370 S$=LEFT$(S$,E*32+1)+T$+MID$(S$,E*32+3)
380 REM * TORLES ES VISSZAIRAS
390 PRINT#15,"B-P 2 0"T;S
400 PRINT#2,S$;
410 PRINT#15,"U2 2 0"T;S
420 CLOSE 2:CLOSE 15
425 PRINT "AZ ADATOK BEOLVASHATOK."
430 PRINT "AZ ERVENYES ADATOK MASOLASA UTAN"
440 PRINT "A KOVETKEZO UTASITASOKAT LEHET MEGADNI:"
450 PRINT
460 PRINT "OPEN 15,8,15"
470 PRINT CHR$(17)"PRINT#15,"CHR$(34)"S:"F$CHR$(34)
480 PRINT CHR$(17)"PRINT#15,"CHR$(34)"V0"CHR$(34)
490 PRINT CHR$(17)"CLOSE 15"
500 END

```

9. FEJEZET

A POKE ÉS MÁS HASZNOS RUTINOK

9.1. A kazettapuffer mint a program tárolására alkalmas hely

Ha valaki egy BASIC programot gépi kódú programmal együtt szeretne használni, el kell döntenie, hogy a tárban hová helyezze a gépi kódú programot. A tárterület kiválasztásánál ügyelni kell arra, hogy azt a BASIC programok vagy változók nehegy felülírják. Vagy olyan területet kell keresnünk, amit a BASIC interpreter nem használ, vagy módosítanunk kell a BASIC kezdő- és végcímét.

A tárban három olyan terület van, amelyet a BASIC nem használ, ezek közül az egyik a kazettapuffer.

A kazettapuffer a 828-as címtől az 1019-es címig helyezkedik el (\$033C – \$03FB). Ezt a területet csak azok a programok használják, amelyek a kazettás egységen dolgoznak. Itt tehát rendelkezésünkre áll 192 byte gépi kódú programok tárolására. Mivel a 13-as, 14-es, ill. 15-ös sprite-ok a kazettapuffert használják, ha olyan programot írunk, amely dolgozik ezekkel a sprite-okkal, erre a területre nem helyezhetünk gépi kódú programot. A 11-es sprite adatait a rendszer a 704-től 767-ig (\$02C0, \$02FF) terjedő 64 byte-nyi területre helyezi. Ha a 11-es sprite-ra nincs szükségünk, ezzel a területtel is szabadon gazdálkodhatunk. A BASIC interpreter fölött, 49152-től 53247-ig (\$C000, \$CFFF) terjedő 4 kbyte-os szabad RAM területen általában hosszabb gépi kódú programokat szoktunk elhelyezni.

Ha csak néhány szabad tárcímre van szükségünk, pl. olyan változók tárolására, amelyek értékét a CLR és a NEW utasítások végrehajtása után is meg akarjuk őrizni, érdemes a képernyőtár „mögött” található 16 szabad byte-ot felhasználni. A képernyőtár 1 kbyte = 1024 byte, a videoram pedig további $25 \times 40 = 1000$ byte területet foglal el a tárban. A második 1 kbyte-ból tehát 24 szabad byte marad, amelyből 8 byte-ot a sprite-mutatók lekötnek. A maradék 16 byte a 2024-es címtől a 2039-es címig (\$07E9, \$07F8) a programozó rendelkezésére áll.

Előfordulhat, hogy olyan programot írtunk, amely olyan sok gépi kódú részt tartalmaz, hogy azok nem férnek el a fent ismertetett szabad területeken. Ez még mindig nem jelenti a feladat megoldhatatlanságát.

A BASIC területet ugyanis tetszőlegesen leszűkíthetjük néhány POKE utasítással, és az így kapott szabad területtel ismét szabadon gazdálkodhatunk.

A BASIC interpreter a 43/44-es (\$2B/\$2C), ill. 55/56-os (\$37/\$38) címeken tárolja a BASIC program kezdetének és végének mutatóit. Ezeket a mutatókat az alábbi utasításokkal írhatjuk ki képernyőre:

```
PRINT PEEK (43) + 256*PEEK (44) ill.
```

```
PRINT PEEK (55) + 256*PEEK (56)
```

Az eredmény (ha az alapértelmezést nem változtattuk meg) 2049, ill. 40960. Tegyük fel, hogy 1000 byte-ot akarunk felszabadítani, tehát a BASIC terület végét 1000 byte-tal lejjebb kell helyezni, azaz a mutatót 39960-ra kell beállítani. A módosító utasítások:

HB = INT (39960/256) : LB = 39960 - HB*256

POKE 55, LB : POKE 56, HB : CLR

A CLR utasításra feltétlenül szükség van, egyébként ugyanis a programban használt változók értéke helytelen lenne. Hasonlóan módosíthatjuk a BASIC terület kezdetét is:

HB = INT (3049/256) : LB = 3049 - HB*256

POKE 43, LB : POKE 44, HB : POKE 3049-1, 0 : NEW

A NEW utasítás az összes BASIC mutatót helyreállítja.

9.2 A füzérek rendezése

Gyakori programozói feladat az adatok, számsorok, nevek, címek stb. rendezése. Az irodalomban nagyon sok rendező algoritmust találhatunk, azonban nagy adathalmaz esetében ezek többnyire lassúak. Minél könnyebben érthető az algoritmus, általában annál lassabb a rendezés. Ha valóban gyors rendezésre van szükségünk, a programot gépi nyelven kell megírni. Az összehasonlítás azt mutatja, hogy ugyanazt az algoritmust – ugyanazt a feladatot – 100-szor gyorsabban valósíthatjuk meg gépi kódban megírt programmal, mint BASIC nyelvűvel. A következő gépi kódú programmal szöveges változókat rendeztünk abc-be.

Használatának feltételei:

1. A rendezendő füzértömböt DIM utasítással kell deklarálnunk.

2. A tömb utolsó elemébe üres karaktert kell tenni.

Az üres füzérre azért van szükség, mert a tömb számára lefoglalt helyet nem minden esetben használjuk ki teljesen. A program futási idejét az üres karakterek rendezése feleslegesen és alaposan megnövelné. Az általunk bemutatott program figyel, hogy egy tömbelem üres karaktert tartalmaz-e, és ha igen, ezt tekinti a tömb utolsó elemének.

Mint hogy a program rövid, a kazettapufferbe helyeztük, így a SYS 828 utasítással hívható.

Híváskor ellenőrzi, hogy a BASIC program első tömbje egyindexes füzérváltozó-e vagy sem; ha nem, a végrehajtást megszakítja.

A gépi kódú program listája:

```

0 REM **** P75. ****
1 :
2 :
100 : 033C      *= 828
105 : 033C      .OPT P1
110 : 033C      LDY #0
120 : 033E      LDA ($2F),Y ;AZ ELSO BETU
130 : 0340      BMI L1
140 : 0342      INY
150 : 0343      LDA ($2F),Y ;A MASODIK KARAKTER
160 : 0345      BPL L1
170 : 0347      LDY #4
180 : 0349      LDA ($2F),Y ;DIMENZIOK
190 : 034B      CMP #1
200 : 034D      BEQ L2
210 : 034F      L1 RTS
220 : 0350      L2 CLC
230 : 0351      LDA $2F      ;TOMBKEZDET
240 : 0353      ADC #7        ; PLUSZ 7
250 : 0355      STA $6E
260 : 0357      LDA $30
270 : 0359      ADC #0
280 : 035B      STA $6F
290 : 035D      L3 LDY #0
300 : 035F      LDA ($6E),Y
310 : 0361      BEQ L1        ;A HOSSZ NULLA,KESZ
320 : 0363      STA $22
330 : 0365      L4 INY
340 : 0366      LDA ($6E),Y
350 : 0368      STA $22,Y     ;MUTATO A FUZERRE
360 : 036B      CPY #2
370 : 036D      BNE L4
380 : 036F      LDA $6E
390 : 0371      STA $71
400 : 0373      LDA $6F
410 : 0375      STA $72
420 : 0377      L5 CLC
430 : 0378      LDA $71

```

```

440 : 037A 69 03      ADC #3      ;HAROM HOZZAADASA
450 : 037C 85 71      STA $71
460 : 037E 90 02      BCC L6
470 : 0380 E6 72      INC $72
480 : 0382 A0 00      LDY #0
490 : 0384 B1 71      LDA ($71),Y
500 : 0386 F0 3D      BEQ L13
510 : 0388 85 4D      STA $4D      ;HOSSZ
520 : 038A C5 22      CMP $22      ;OSSZEHASONLITAS AZ ELSO HOSSZAL
530 : 038C 90 02      BCC L7
540 : 038E A5 22      LDA $22
550 : 0390 85 55      STA $55      ;A HOSSZAK OSSZEHASONLITASA
560 : 0392 C8        L8      INY
570 : 0393 B1 71      LDA ($71),Y
580 : 0395 99 4D 00    STA $4D,Y
590 : 0398 C0 02      CPY $2
600 : 039A D0 F6      BNE L8
610 : 039C A0 00      LDY #0
620 : 039E B1 23      L9      LDA ($23),Y
630 : 03A0 D1 4E      CMP ($4E),Y      ;A FUZEREK OSSZEHASONLITASA
640 : 03A2 F0 04      BEQ L10      ;HA EGYENLOEK,AKKOR TOVABB
650 : 03A4 B0 0B      BCS L11      ;HA NAGYOBB,AKKOR CSERE
660 : 03A6 90 CF      BCC L5      ;HA KISEBB,AKKOR ATLEPES A
665 :      ;KOVETKEZO FUZERRE
670 : 03A8 C8        L10     INY
680 : 03A9 C4 55      CPY $55      ;AZ OSSZES KARAKTER EGYENLO
690 : 03AB D0 F1      BNE L9
700 : 03AD C4 22      CPY $22      ;AZ ELSO FUZER HOSSZABB
710 : 03AF B0 C6      BCS L5      ;HA NEM ,AKKOR OK.
720 : 03B1 A0 02      L11     LDY #2
730 : 03B3 B1 6E      L12     LDA ($6E),Y      ;A FUZERMUTATOK CSEREJE
740 : 03B5 AA        TAX
750 : 03B6 B1 71      LDA ($71),Y
760 : 03B8 91 6E      STA ($6E),Y
770 : 03BA 99 22 00    STA $22,Y
780 : 03BD 8A        TXA
790 : 03BE 91 71      STA ($71),Y
800 : 03C0 80        DEY
810 : 03C1 10 F0      BPL L12
820 : 03C3 30 B2      BMI L5
830 : 03C5 18        L13     CLC      ;MUTATO A KOVETKEZO FUZERRE
840 : 03C6 A5 6E      LDA $6E
850 : 03C8 69 03      ADC #3
860 : 03CA 85 6E      STA $6E
870 : 03CC 90 8F      BCC L3
880 : 03CE E6 6F      INC $6F
890 : 03D0 D0 8B      BNE L3

```

READY.

A BASIC betöltőprogram:

```

0 REM **** P76. ****
1 :
2 :
100 FOR I=028 TO 977
110 READ X:POKE I,X:S=S+X:NEXT
120 DATA 160, 0,177, 47, 48, 13,200,177, 47, 16, 8,160
130 DATA 4,177, 47,201, 1,240, 1, 96, 24,165, 47,105
140 DATA 7,133,110,165, 48,105, 0,133,111,160, 0,177
150 DATA 110,240,236,133, 34,200,177,110,153, 34, 0,192
160 DATA 2,208,246,165,110,133,113,165,111,133,114, 24
170 DATA 165,113,105, 3,133,113,144, 2,230,114,160, 0
180 DATA 177,113,240, 61,133, 77,197, 34,144, 2,165, 34
190 DATA 133, 85,200,177,113,153, 77, 0,192, 2,208,246

```

```

200 DATA 160, 0,177, 35,209, 78,240, 4,176, 11,144,207
210 DATA 200,196, 85,208,241,196, 34,176,198,160, 2,177
220 DATA 110,170,177,113,145,110,153, 34, 0,138,145,113
230 DATA 136, 16,240, 48,178, 24,165,110,105, 3,133,110
240 DATA 144,143,230,111,208,139
250 IF S<>17663 THEN PRINT "HIBA A DATA SORBAN..!":END
260 PRINT "OK..!!"

```

A gépi kódú rutin végrehajtási sebességét az alábbi BASIC programmal szeretnénk bemutatni, amely egy füzéreket tartalmazó tömböt rendez, majd a rendezett sort, ill. a rendezési időt kiírja a képernyőre.

A füzerek számát és hosszát előre megadhatjuk.

```

0 REM **** P77. ****
1 :
2 :
100 INPUT "SZAM, HOSSZ ";A,L
110 DIM A$(A):A=A-1
120 FOR I=0 TO A
130 FOR J=1 TO RND(1)*L
140 A$(I)=A$(I)+CHR$(RND(1)*26+65)
150 NEXT J:NEXT I
160 FOR I=0 TO A:PRINT A$(I):NEXT I
170 T=TI:SYS 828:T=TI-T
180 PRINT "RENDEZESI IDO "T/60" MASODPERC"
190 FOR I=0 TO A:PRINT A$(I):NEXT I

```

Próbáljuk ki a programot különböző számú és hosszúságú füzerváltozókkal. Azt fogjuk tapasztalni, hogy 100 füzér rendezését kevesebb, mint egy másodperc alatt végrehajtja, szemben egy BASIC programmal, amelynek hasonló feladat végrehajtásához általában legalább egy percre van szüksége.

A gépi kódú program használatakor ne feledkezzünk meg az üres füzerről!

9.3 Az indexes változó legkisebb és legnagyobb értéke

Gyakori feladat egy indexes változó legkisebb és legnagyobb értékének meghatározása. Ezt viszonylag egyszerűen, egy kis BASIC ciklussal meg tudjuk oldani. A ciklus lefutásához szükséges idő az elemek számával meglehetősen gyorsan nő.

Tipikus feladat, amelyre célszerű gépi kódot rutint írni! Az algoritmus ugyanaz mint a BASIC program esetében, de a végrehajtási idő nagyságrenddel csökken.

```
0 REM **** P78. ****
1 :
2 :
100 DIM A(N)
200 GOSUB 1000
1000 MIN = A(0)
1010 FOR I=1 TO N
1020 IF A(I) < MIN THEN MIN = A(I)
1030 NEXT
1040 RETURN
```

A 100-as sorban az A indexes változót N felső indexhatárral deklaráltuk. Az 1000-es sorban kezdődő szubrutinban a 1010-es sorban keressük a legkisebb elemet. Ha a legnagyobb elemet akarjuk megkeresni, cseréljük fel a 1020-as sort az alábbira:

```
1020 IF A(I) > MAX THEN MAX = A(I)
```

az 1000-es sorba pedig MAX = A(0)-t írjunk a MIN = A(0) helyett.

A gépi kódú rutin nemcsak gyorsabb, mint a BASIC program, hanem paraméterezhető is. Hívásakor USR függvény-paraméterként átadhatjuk BASIC-ből az indexes változó nevét. A numerikus tömb tartalmazhat valós vagy egész típusú (A%) számokat.

A programot a \$C800-as címtől kezdve helyeztük el a tárban a szabad RAM területre.

```
0 REM **** P79. ****
1 :
2 :
110 : ; MIN/MAX FUGGVENY
120 : C800 INTFLG = 14 ;AZ EGESZ TIPUSU VALTOZOK KAPCSOLOJA
130 : C800 STORE = $26
140 : C800 ARRTAB = $2F ;MUTATO A TOMB TABLAZATRA
150 : C800 ARREND = $31 ;MUTATO A TOMBOK VEGERE
160 : C800 VARNAM = $45 ;VALTOZONEV
170 : C800 TEMP = $5F
180 : C800 SETARR = $B196 ;MUTATO A TOMB ELSO ELEMERE
190 : C800 MEMFAC = $BBA2 ;A KONSTANS BETOLTESE A FAC-BA
200 : C800 COMPARE = $BC5B ;A KONSTANSOK OSSZEHASONL. A FAC-AL
210 : C800 ERROUT = $A445
220 : C800 INT = $14 ;AZ EGESZ TIPUSU VALTOZOK CIME
230 : C800 INTFLOAT = $B391 ;AZ EGESZ RESZ FAC-BA
240 : C800 * = $C800
300 : C800 A6 2F MINMAX LDX ARRTAB
310 : C802 A5 30 LDA ARRTAB+1 ;MUTATO A TOMB TABLAZAT KEZDETERE
320 : C804 86 5F L3 STX TEMP
330 : C806 85 60 STA TEMP+1 ;CIKLUSVALTOZO
340 : C808 C5 32 CMP ARREND+1
350 : C80A D0 04 BNE L1
360 : C80C E4 31 COX ARREND ;VEGE A TOMB TABLAZATNAK
370 : C80E F0 1D BEQ NOTFOUND
380 : C810 A0 00 L1 LDY #0
```

| | | | | | | |
|--------|------|----------|----------|-----|----------|-----------------------------------|
| 390 : | C812 | B1 5F | | LDA | (TEMP),Y | ;A NEV ELSO BETUJE |
| 400 : | C814 | C8 | | INY | | |
| 410 : | C815 | C5 45 | | CMP | VARNAM | ;OSSZEHASONLITAS A KERESETT NEVEL |
| 420 : | C817 | D0 06 | | BNE | L2 | |
| 430 : | C819 | A5 46 | | LDA | VARNAM+1 | |
| 440 : | C81B | D1 5F | | CMP | (TEMP),Y | ;A 2.KARAKTER OSSZEHASONLITASA |
| 450 : | C81D | F0 17 | | BEQ | FOUND | ;MEGVAN |
| 460 : | C81F | C8 | L2 | INY | | |
| 470 : | C820 | B1 5F | | LDA | (TEMP),Y | |
| 480 : | C822 | 18 | | CLC | | |
| 490 : | C823 | 65 5F | | ADC | TEMP | ;A REL.CIM (OFFSET) HOZZAADASA |
| 500 : | C825 | AA | | TAX | | |
| 510 : | C826 | CB | | INY | | |
| 520 : | C827 | B1 5F | | LDA | (TEMP),Y | |
| 530 : | C829 | 65 60 | | ADC | TEMP+1 | |
| 540 : | C82B | 90 D7 | | BCC | L3 | |
| 550 : | C82D | A9 88 | NOTFOUND | LDA | #<ERRMSG | ;MUTATO A HIBAUZENETRE |
| 560 : | C82F | 85 22 | | STA | \$22 | |
| 570 : | C831 | A9 C8 | | LDA | #>ERRMSG | |
| 580 : | C833 | 4C 45 A4 | | JMP | ERROUT | ;A HIBAUZENET KIIRASA |
| 590 : | C836 | CB | FOUND | INY | | |
| 600 : | C837 | 18 | | CLC | | |
| 610 : | C838 | B1 5F | | LDA | (TEMP),Y | |
| 620 : | C83A | 65 5F | | ADC | TEMP | |
| 630 : | C83C | 85 26 | | STA | STORE | |
| 640 : | C83E | C8 | | INY | | |
| 650 : | C83F | B1 5F | | LDA | (TEMP),Y | |
| 660 : | C841 | 65 60 | | ADC | TEMP+1 | |
| 670 : | C843 | 85 27 | | STA | STORE+1 | ;MUTATO A TOMB VEGERE |
| 680 : | C845 | C8 | | INY | | |
| 690 : | C846 | B1 5F | | LDA | (TEMP),Y | ;DIMENZIOSZAM |
| 700 : | C848 | 20 96 B1 | | JSR | SETARR | ;MUTATO AZ ELSO TOMBELEMRE |
| 710 : | C848 | 85 5F | | STA | TEMP | |
| 720 : | C84D | 84 60 | | STY | TEMP+1 | ;A MUTATO TAROLASA |
| 730 : | C84F | 24 8E | | BIT | INTFLG | ;A TIPUS ELLENORZESE |
| 740 : | C851 | 30 24 | | BMI | INTEGER | |
| 750 : | C853 | 10 09 | | BPL | LP1 | |
| 760 : | C855 | 20 5B | BC L5 | JSR | COMPARE | ;A TOMBELEM OSSZEHASONLITASA |
| 770 : | C858 | 10 07 | | BPL | LOOP | |
| 780 : | C85A | A5 5F | | LDA | TEMP | |
| 790 : | C85C | A4 60 | | LDY | TEMP+1 | |
| 800 : | C85E | 20 A2 | BB LP1 | JSR | MEMFAC | ;A TOMBELEM TAROLASA MIN/MAX-KENT |
| 810 : | C861 | 18 | LOOP | CLC | | |
| 820 : | C862 | A5 5F | | LDA | TEMP | |
| 830 : | C864 | 69 05 | | HDC | #5 | ;MUTATO A KOVETKEZO ELEMRE |
| 840 : | C866 | 85 5F | | STA | TEMP | |
| 850 : | C868 | 90 02 | | BCC | L4 | |
| 860 : | C86A | E6 60 | | INC | TEMP+1 | |
| 870 : | C86C | A4 60 | L4 | LDY | TEMP+1 | |
| 880 : | C86E | C5 26 | | CMP | STORE | ;A TOMB VEGE |
| 890 : | C870 | D0 E3 | | BNE | L5 | |
| 900 : | C872 | C4 27 | | CPY | STORE+1 | |
| 910 : | C874 | D0 DF | | BNE | L5 | |
| 920 : | C876 | 60 | | RTS | | |
| 930 : | C877 | A0 00 | INTEGER | LDY | #0 | ;EGESZ TIPUSU TOMB |
| 940 : | C879 | B1 5F | | LDA | (TEMP),Y | |
| 950 : | C87B | AA | | TAX | | |
| 960 : | C87C | C8 | | INY | | |
| 970 : | C87D | B1 5F | | LDA | (TEMP),Y | |
| 980 : | C87F | 85 15 | | STA | INT+1 | ;AZ ELSD ERTEK BETOLTESE INT-BE |
| 990 : | C881 | 86 14 | | STX | INT | |
| 1000 : | C883 | 18 | L12 | CLC | | |
| 1010 : | C884 | A5 5F | | LDA | TEMP | |
| 1020 : | C886 | 69 02 | | ADC | #2 | ;MUTATO A KOVETKEZO ELEMRE |
| 1030 : | C888 | 85 5F | | STA | TEMP | |
| 1040 : | C88A | 90 02 | | BCC | L10 | |
| 1050 : | C88C | E6 60 | | INC | TEMP+1 | |

```

1060 : C88E C5 26      L10    CMP STORE
1070 : C890 D0 0D      BNE L11
1080 : C892 A5 60      LDA TEMP+1
1090 : C894 C5 27      CMP STORE+1 ;VEGE VAN-E
1100 : C896 D0 07      BNE L11
1110 : C898 A5 14      LDA INT ;AZ EGESZ ERTEK BETOLTESE
1120 : C89A A4 15      LDY INT+1
1130 : C89C 4C 91 B3    JMP INTFLOAT ;KONVERTALAS
1140 : C89F A0 00      LDY #0
1150 : C8A1 B1 5F      LDA (TEMP),Y
1160 : C8A3 C5 14      CMP INT ;A FELSO BYTE OSSZEHASONLITASA
1170 : C8A5 D0 07      BNE L14
1180 : C8A7 C8          INY
1190 : C8A8 B1 5F      LDA (TEMP),Y
1200 : C8AA E5 15      SBC INT+1 ;AZ ALSO BYTE OSSZEHASONLITASA
1210 : C8AC F0 D5      BEQ L12
1220 : C8AE A9 01      LDA #1 ;A 'NAGYOBB' KAPCSOLO
1230 : C8B0 90 02      BCC L13
1240 : C8B2 A9 FF      LDA #$FF ;A 'KISEBB' KAPCSOLO
1250 : C8B4 30 C1      L13    BMI INTEGER
1260 : C8B6 10 CB      BPL L12
1270 : C8B8 41 52 52 ERRMSG .ASC "ARRAY NOT FOUND"
1280 : C8BB 41 59 20 4F 54 20 46 50 55 4F C4

```

A BASIC betöltőprogram:

```

0 REM **** P80. ****
1 :
2 :
100 FOR I=51200 TO 51398
110 READ X:POKEI,X:S=S+X:NEXT
120 DATA 166, 47,165, 48,134, 95,133, 96,197, 50,208, 4
130 DATA 228, 49,240, 29,160, 0,177, 95,200,197, 69,208
140 DATA 6,165, 70,209, 95,240, 23,200,177, 95, 24,101
150 DATA 95,170,200,177, 95,101, 96,144,215,169,184,133
160 DATA 34,169,200, 76, 69,164,200, 24,177, 95,101, 95
170 DATA 133, 38,200,177, 95,101, 96,133, 39,200,177, 95
180 DATA 32,150,177,133, 95,132, 96, 36, 14, 48, 36, 16
190 DATA 9, 32, 91,188, 16, 7,165, 95,164, 96, 32,162
200 DATA 187, 24,165, 95,105, 5,133, 95,144, 2,230, 96
210 DATA 164, 96,197, 38,208,227,196, 39,208,223, 96,160
220 DATA 0,177, 95,170,200,177, 95,133, 21,134, 20, 24
230 DATA 165, 95,105, 2,133, 95,144, 2,230, 96,197, 38
240 DATA 208, 13,165, 96,197, 39,208, 7,165, 20,164, 21
250 DATA 76,145,179,160, 0,177, 95,197, 20,208, 7,200
260 DATA 177, 95,229, 21,240,213,169, 1,144, 2,169,255
270 DATA 48,193, 16,203, 65, 82, 82, 65, 89, 32, 78, 79
280 DATA 84, 32, 70, 79, 85, 78,196
290 IF S<>22908 THEN PRINT "HIBA A DATA SORBAN!!":END
300 PRINT "OK!"

```

Ez a változat az adott tömb legnagyobb elemét keresi meg, de könnyen módosíthatjuk úgy, hogy a legnagyobb elem helyett a legkisebb elemet keresse meg.

A szükséges módosítások:

- a C858-ra a \$10 helyett \$30-at
- a C8B4-re a \$30 helyett \$10-et
- a C8B6-ra a \$10 helyett \$30-at

Hívás előtt az USR függvény címét is be kell állítani.

POKE 785,0 : POKE 786,200

A USR függvényt az utasításfüggvényekhez hasonlóan használhatjuk, és argumentuma egész típusú változó is lehet, pl.:

$X = \text{USR}(A\%) * \text{SIN}(3)$

Próbáljuk ki a gépi kódú rutint az alábbi BASIC programmal:

```
0 REM **** FB1. ****
1 :
2 :
100 POKE 785,0:POKE 786,200
110 INPUT "TOMBMERET ";N
120 DIM A(N)
130 FOR I=1 TO N
140 A(I)=RND(1)*1000
150 PRINT A(I)
160 NEXT
170 PRINT
180 PRINT USR(A)
```

Az átkapcsolás a legnagyobb elem kereséséről a legkisebb elem keresésére és megfordítva:

POKE 51288,48 ill. 16

POKE 51380,16 ill. 48

POKE 51382,48 ill. 16

A program futtatása során figyeljük meg a végrehajtási sebességben történő változást valós, ill. egész típusú tömb esetén. (A különbség csak viszonylag nagy méretű tömböknél érzékelhető pl.: DIM A%(5000).)

9.4 A DUMP utasítás – a változók tartalmának kiírása

Az alábbi gépi kódú program, amelyet a kazettapufferbe helyeztünk, kiválóan alkalmas BASIC programok tesztelésére. Hívásakor kiírja a BASIC programban előforduló változókat, értékeikkel együtt.

```
0 REM **** P82. ****
1 :
2 :
100 : 033C          * = 828      ; KAZETTAPUFFER
110 : 033C          LDA $2D
120 : 033E          LDY $2E      ; A VALTOZOKEZDET-MUTATO
130 : 0340          STA $14      ; TAROLASA
140 : 0342          STY $15
150 : 0344          CPY $30      ; OSSZEHASONLITASA A VALTOZOSZOVEG
155 :              ; MUTATOVAL
160 : 0346          D0 02        BNE L1
170 : 0348          C5 2F        CMP $2F
180 : 034A          B0 18        L1      BCS L3      ; UGRAS A VEGERE, HA KESZ
190 : 034C          69 02        ADC #2      ; MUTATO A VALTOZOERTEKRE
200 : 034E          90 01        BCC L2
210 : 0350          C8          INY
220 : 0351          85 22        L2      STA $22
230 : 0353          84 23        STY $23
240 : 0355          20 82 03      JSR L7      ; A NEV KIIRASA
250 : 0358          20 B6 03      JSR L12     ; AZ "=" KIIRASA
260 : 035B          8A          TXA
270 : 035C          10 07        BPL L4
280 : 035E          20 BF 03      JSR L13     ; AZ EGESZTIPUSU VALTOZO KIIRASA
290 : 0361          4C 71 03      JMP L6      ; UGRAS A FOCIKLUSHOZ
300 : 0364          60          L3      RTS
310 : 0365          98          L4      TYA
320 : 0366          30 06        BMI L5
330 : 0368          20 CF 03      JSR L14     ; A LEBEGOPONTOS SZAM KIIRASA
340 : 036B          4C 71 03      JMP L6
350 : 036E          20 D8 03      L5      JSR L16     ; A FUZERVALTOZO KIIRASA
360 : 0371          A9 0D        L6      LDA #13     ; KOCSI VISSZA
370 : 0373          20 D2 FF      JSR $FFD2    ; KIIRASA
380 : 0376          A5 14        LDA $14
390 : 0378          A4 15        LDY $15
400 : 037A          18          CLC
410 : 037B          69 07        ADC #7      ; A KOVETKEZO VALTOZO +7
420 : 037D          90 C1        BCC L0
430 : 037F          C8          INY
440 : 0380          B0 BE        BCS L0      ; VISSZA A FOCIKLUSHOZ
450 : 0382          A0 00        LDY #0
460 : 0384          B1 14        L7      LDA ($14),Y ; A NEV ELSO BETUJENEK
470 : 0386          AA          TAX
480 : 0387          29 7F        AND #$7F
490 : 0389          20 D2 FF      JSR $FFD2    ; KIIRASA
500 : 038C          C8          INY
510 : 038D          B1 14        LDA ($14),Y ; A KOVETKEZO KARAKTER
520 : 038F          A8          TAY
530 : 0390          29 7F        AND #$7F
540 : 0392          F0 03        BEQ L8
550 : 0394          20 D2 FF      JSR $FFD2    ; KIIRASA
560 : 0397          8A          L8      TXA
570 : 0398          10 11        BPL L10     ; A TIPUS ELLENORZESE
580 : 039A          98          TYA
590 : 039B          30 0A        BMI L9
600 : 039D          A9 21        LDA #"!"     ; FUGGVENY "!" KIIRASA
610 : 039F          20 D2 FF      JSR $FFD2
620 : 03A2          68          PLA
630 : 03A3          68          PLA
```


| | | | | | | | | |
|------|---|------|--------------|-----|-----|----------|--|----------------------------------|
| 640 | : | 03A4 | 4C 71 03 | | JMP | L6 | | ;VISSZAUGRAS A FOCIKLUSHOZ |
| 650 | : | 03A7 | A9 25 | L9 | LDA | #"%" | | ;EGESZTIPUSU VALTOZO |
| 660 | : | 03A9 | D0 4E | | BNE | L19 | | |
| 670 | : | 03AB | 98 | L10 | TYA | | | |
| 680 | : | 03AC | 10 04 | | BPL | L11 | | |
| 690 | : | 03AE | A9 24 | | LDA | #"%" | | ;FUZERVALTOZO |
| 700 | : | 03B0 | D0 47 | | BNE | L19 | | |
| 710 | : | 03B2 | 60 | L11 | RTS | | | |
| 720 | : | 03B3 | 20 D2 FF | | JSR | #FFD2 | | ;A KARAKTER KIIRASA |
| 730 | : | 03B6 | A9 20 | L12 | LDA | #" " | | |
| 740 | : | 03B8 | 20 D2 FF | | JSR | \$FFD2 | | ;URES KARAKTER KIIRASA |
| 750 | : | 03BB | A9 30 | | LDA | #"=" | | |
| 760 | : | 03BD | D0 3A | | BNE | L19 | | ;KIIRAS |
| 770 | : | 03BF | A0 00 | L13 | LDY | #0 | | ;AZ EGESZ TIPUSU VALTOZO |
| 780 | : | 03C1 | B1 22 | | LDA | (\$22),Y | | ;ALSO BYTE |
| 790 | : | 03C3 | AA | | TAX | | | |
| 800 | : | 03C4 | C8 | | INY | | | |
| 810 | : | 03C5 | B1 22 | | LDA | (\$22),Y | | ;FELSO BYTE |
| 820 | : | 03C7 | A8 | | TAY | | | |
| 830 | : | 03C8 | 8A | | TXA | | | |
| 840 | : | 03C9 | 20 95 B3 | | JSR | #B395 | | ;ATVALTAS LEBEGOPONTOSSA |
| 850 | : | 03CC | 4C D2 03 | | JMP | L15 | | ;EB KIIRAS |
| 860 | : | 03CF | 20 A6 BB L14 | | JSR | #BBA6 | | ;A LEBEGOPONTOS VALTOZO BETOLTES |
| 870 | : | 03D2 | 20 DD BD L15 | | JSR | #BDDD | | ;AZ ASCII-FUZER ATVALTASA |
| 880 | : | 03D5 | 4C 1E AB | | JMP | #AB1E | | ;KIIRASA |
| 890 | : | 03D8 | 20 F7 03 L16 | | JSR | L18 | | ;FUZER,FELSOVESSZO KIIRASA |
| 900 | : | 03DB | A0 02 | | LDY | #2 | | |
| 910 | : | 03DD | B1 22 | | LDA | (\$22),Y | | ;A CIM (FELSO BYTE) |
| 920 | : | 03DF | B5 25 | | STA | #25 | | |
| 930 | : | 03E1 | 88 | | DEY | | | |
| 940 | : | 03E2 | B1 22 | | LDA | (\$22),Y | | ;A CIM (ALSO BYTE) |
| 950 | : | 03E4 | B5 24 | | STA | #24 | | |
| 960 | : | 03E6 | 88 | | DEY | | | |
| 970 | : | 03E7 | B1 22 | | LDA | (\$22),Y | | ;HOSSZ |
| 980 | : | 03E9 | B5 26 | | STA | #26 | | |
| 990 | : | 03EB | F0 0A | | BEQ | L18 | | |
| 1000 | : | 03ED | B1 24 | L17 | LDA | (\$24),Y | | ;A FUZER KARAKTEREINEK |
| 1010 | : | 03EF | 20 D2 FF | | JSR | #FFD2 | | ;KIIRASA |
| 1020 | : | 03F2 | C8 | | INY | | | |
| 1030 | : | 03F3 | C4 26 | | CPY | #26 | | ;A FUZER VESE |
| 1040 | : | 03F5 | D0 F6 | | BNE | L17 | | |
| 1050 | : | 03F7 | A9 22 | L18 | LDA | ##22 | | ;FELSOVESSZO |
| 1060 | : | 03F9 | 4C D2 FF L19 | | JMP | #FFD2 | | ;KIIRASA |

A gépi kódú program BASIC betöltője:

```

0 REM **** PB3. ****
1 :
2 :
100 FOR I=828 TO 1019
110 READ X:POKE I,X: S=S+X: NEXT I
120 DATA 165, 45,164, 46,133, 20,132, 21,196, 48,208, 2
130 DATA 197, 47,176, 24,105, 2,144, 1,200,133, 34,132
140 DATA 35, 32,130, 3, 32,182, 3,138, 16, 7, 32,191
150 DATA 3, 76,113, 3, 96,152, 48, 6, 32,207, 3, 76
160 DATA 113, 3, 32,216, 3,169, 13, 32,210,255,165, 20
170 DATA 164, 21, 24,105, 7,144,193,200,176,190,160, 0
180 DATA 177, 20,170, 41,127, 32,210,255,200,177, 20,168
190 DATA 41,127,240, 3, 32,210,255,138, 16, 17,152, 48
200 DATA 10,169, 33, 32,210,255,104,104, 76,113, 3,169
210 DATA 37,208, 78,152, 16, 4,169, 36,208, 71, 96, 32
220 DATA 210,255,169, 32, 32,210,255,169, 61,208, 58,160
230 DATA 0,177, 34,170,200,177, 34,168,138, 32,149,179
240 DATA 76,210, 3, 32,166,187, 32,221,189, 76, 30,171
250 DATA 32,247, 3,160, 2,177, 34,133, 37,136,177, 34

```

```

260 DATA 133, 36,136,177, 34,133, 38,240, 10,177, 36, 32
270 DATA 210,255,200,196, 38,208,246,169, 34, 76,210,255
280 IF S<>20988 THEN PRINT "HIBA A DATA SORBAN..!!":END
290 PRINT "OK...!!"

```

Futtassuk le az alábbi programot:

```

0 REM **** PB4. ****
1 :
2 :
100 A=5
110 DEF FN(X)=SIN(X)*COS(X)
120 C$="PROGRAM"
130 B%=-101
140 SYS828

```

A futtatás eredménye:

```

A = 5
X !
Y = 0
C$ = "PROGRAM"
B% = -101

```

A SYS 828 utasítás hatására megjelennek a képernyőn a BASIC programban szereplő változók. Természetesen közvetlenül, parancsüzemmódban is meghívhatjuk a rutint. Ha egy programot futás közben megszakítunk, a SYS 828 utasítással kiírathatjuk a változókat, majd CONT utasítással folytathatjuk a program futtatását, ami nagyon hasznos a BASIC program tesztelési fázisában.

(A képernyőn a változó neve mögött kiírt felkiáltójel arra utal, hogy ez a változó függvénydefiniációban szerepel.)

9.5 A módosított PEEK függvény

A következő gépi kódú program megkönnyíti a hozzáférést a Commodore 64-es RAM területéhez. A módosított PEEK függvénnyel a karaktergenerátor területéről is tudunk olvasni.

A \$A00-tól \$BFFF-ig (40960, 49151) és a \$E000-tól \$FFFF-ig (57344, 65535) terjedő területek a tárban kétszeresen foglaltak: itt található a BASIC ROM, ill. a Kernal ROM, vagy kétszer 8K RAM. Ezt a 16 kbyte-os területet a BASIC mégsem használhatja minden további nélkül. Minden erre a területre vonatkozó POKE utasítás a RAM területre ír, ugyanakkor minden idevonatkozó PEEK utasítás a bekapcsolt ROM területéről olvas. Helyettesítsük a PEEK függvényt egy olyan saját USR függvénnyel, amely a ROM területéről olvas. Mielőtt a tárcím tartalmát kiolvasnánk, átkapcsoljuk ezt a területet RAM állapotra, majd olvasás után visszakapcsolunk ROM állapotra. A karaktergenerátor a \$D000 tárcímától a \$DFFF tárcímig terjedő területen található. Ahhoz, hogy innen olvasni tudjunk, meg kell vizsgálni a tárcímet, hogy belesik-e ebbe a tartományba. Ha igen, átkapcsolunk RAM állapotra, majd a tárcím tartalmát kiolvasva ismét visszakapcsolunk.

```
0 REM **** P85. ****
1 :
2 :
100 :                ; USR - PEEK
110 : 033C          ADR = $14      ; EGESZ-CIM
120 : 033C          FACADR = $B7F7
130 : 033C          YFAC = $B3A2
190 : 033C          *= 828      ; KAZETTAPUFFER
200 : 033C A5 14      LDA ADR
210 : 033E 48        PHA          ; AZ EGESZ-CIM TARDLASA
220 : 033F A5 15      LDA ADR+1
230 : 0341 48        PHA
240 : 0342 20 F7 B7   JSR FACADR  ; A FAC ATVALTASA CIM FORMATUMRA
250 : 0345 A5 01      LDA 1
260 : 0347 48        PHA          ; A KONFIGURACIO TARDLASA
261 : 0348 A5 15      LDA ADR+1
262 : 034A C9 D0      CMP #$D0    ; KISEBB-E, MINT A $D000
263 : 034C 90 07      BCC RAM
264 : 034E C9 E0      CMP #$E0    ; NAGYOBB-E, MINT A $E000
265 : 0350 B0 03      BCS RAM
270 : 0352 A9 31      LDA #$31    ; A KARAKTERGENERATOR KIOLVASASA
280 : 0354 2C        .BYT $2C
290 : 0355 A9 34      RAM LDA #$34 ; A RAM KIOLVASASA
300 : 0357 78        SEI
310 : 0358 85 01      STA 1      ; A TARKONFIGURACIO BEALLITASA
320 : 035A A0 00      LDY #0
330 : 035C B1 14      LDA (ADR),Y ; EGY BYTE OLVASASA ES
340 : 035E A8        TAY          ; TARDLASA Y-BAN
350 : 035F 68        PLA
360 : 0360 85 01      STA 1      ; A KONFIGURACIO VISSZATOLTESE
370 : 0362 58        CLI
380 : 0363 68        PLA
390 : 0364 85 15      STA ADR+1
400 : 0366 68        PLA          ; A CIM VISSZATOLTESE
410 : 0367 85 14      STA ADR
420 : 0369 4C A2 B3   JMP YFAC    ; ES KONVERTALASA
```

A fenti kis gépi kódú rutint a 828-as címtől kezdődő kazettapufferbe helyeztük. Betöltés után az USR vektor címét át kell állítani 828-ra:

POKE 785, 828 AND 255
POKE 786, 828/256

Azoknak, akik nem programoznak gépi kódban, ismét közöljük a BASIC betöltőprogramot, amely egyben az USR vektor mutatóját is átállítja.

```
0 REM **** P86. ****
1 :
2 :
100 FOR I=828 TO 875
110 READ X:POKE I,X:S=S+X:NEXT
120 DATA 165, 20, 72,165, 21, 72, 32,247,183,165, 1, 72
130 DATA 165, 21,201,208,144, 7,201,224,176, 3,169, 49
140 DATA 44,169, 52,120,133, 1,160, 0,177, 20,168,104
150 DATA 133, 1, 88,104,133, 21,104,133, 20, 76,162,179
160 IF S<>5085 THEN PRINT "HIBA A DATA SORBAN..!!":END
170 POKE 785,828AND255:POKE 786,828/256
180 PRINT "OK..!!"
```

Az alábbi mintaprogram az USR függvény segítségével visszaadja a karaktermátrix tetszőleges elemét.

```
0 REM **** P87. ****
1 :
2 :
100 CG=13*4096
110 A=(PEEK(53248+24)AND2)*1024
120 INPUT "A KARAKTER KEPERNYO KODJA";B
130 FOR I=1 TO 7
140 PRINT I,USR(CG+A+B*B+I):NEXT
150 GOTO 110
```

A 110-es sor kiválasztja a karaktergenerátor alsó vagy felső felét, attól függően, hogy a nagybetű/grafika vagy kis-/nagybetű üzemmódra van szükség.

Ha most a keresett jelet a képernyő bal felső sarkán beütjük, PEEK (1024) utasítással a kódot lekérdezhetjük. A gépi kódú rutin lehetővé teszi, hogy BASIC programunkhoz az alapállapotban adott tárterületeken kívül 16 kbyte területet használhassunk adattárolásra.

9.6 Multiprogramozás a Commodore 64-esen

A „multiprogramozás” fogalom eredetileg az adatfeldolgozás területéről származik és több program egyidejű futtatását jelenti, egy adott számítógépen. Milyen elven alapul a multiprogramozás?

A legnagyobb számítógép is egyszerre csak egy programot tud futtatni, a multiprogramozás tehát tulajdonképpen nem egyéb, mint ügyes trükk.

Tegyük fel, hogy a számítógép azt a feladatot kapta, hogy egyszerre 5 programot futtasson. Hogyan történik ez valójában?

A gép elindítja az első programot, majd egy bizonyos idő elteltével (ez az idő általában a másodperc tört része) megszakítja ezt a programot, tárolja a változók aktuális tartalmát, és megkezdí a második program feldolgozását. Ugyanannyi idő elteltével ezt a programot is megszakítja, majd veszi a következőt stb.

Amikor ismét az első programra kerül a vezérlés, a gép az első program változóinak tartalmát visszatölti a tárba, így a feldolgozást ott folytathatja, ahol az előző futásnál abbahagyta. Mivel az operációs rendszer ilyenkor a gépidőt megosztja az egyes programok között, mindegyikre az idő tört részét szánva, ezt a módszert gyakran időosztásos (time sharing) programozásnak is nevezik.

Multiprogramozásra, bizonyos korlátozásokkal a Commodore 64-es is képes. A C64-esben tulajdonképpen mindig két program fut egyszerre: a BASIC program és az ún. megszakító-program (interrupt), amelyre a másodperc minden hatvanad részében kerül a vezérlés (ezt a folyamatot egy belső óra irányítja). A megszakítóprogram működése közben a BASIC program végrehajtása szünetel. A megszakítóprogram gondoskodik például a billentyűzet folyamatos lekérdezéséről. A megszakítási időket ügyesen felhasználhatjuk saját céljainkra. Megtehetjük pl., hogy minden ilyen időpillanatban a szövegpufferből egy karaktert küldünk a nyomtatóhoz, tehát miközben a BASIC program fut, a töredék időkből kinyomtathatunk egy szöveget.

Bemutatunk egy olyan mintaprogramot, amely a töredékidőket arra használja, hogy minden tizedmásodpercben kijelzi az aktuális időt a képernyőre. A program a C64-es belső hardver-óráját használja. Természetesen gépi kódban készült, de szokás szerint mellékeljük a BASIC betöltőprogramot is.

```

0 REM **** P88. ****
1 :
2 :
100 : C800
110 : C800
120 : C800
130 : C800
140 : C800
150 : C800
160 : C800
170 : C800
180 : C800
190 : C800
200 : C800
210 : C800
220 : C800
230 : C800
240 : C800
250 : C800
260 : C800
270 : C800
280 : C800

; P88
; TIME, IDŐ KIJELZÉSE
; SYS AD, "HHMMSS", SZIN

FRMEVL = $A09E ; A BASIC-KIFEJEZÉS BETÖLTÉSE
FRESTR = $B6A3
CHKCOM = $AEFD ; VESSZO KERESÉSE
CHRGOT = $79
GETBYT = $B79E ; BYTE-KIFEJEZÉS BETÖLTÉSE
ILLQUAN = $B24B ; 'ILLEGAL QUANTITY'
ADR = $22
COLOR = $2A7 ; SZINTAR
VIDEO = $288 ; A VIDEORAM FELSO BYTE-JA
TEMP = $FB
JRG = $314 ; IRQ-VEKTOR
PNT = $FB
IRGVEC = $EA31 ; IRQ-RUTIN
FARBE = $D800 ; SZINRAM
ZEHNTEL = $DC0B ; VALOSIDEJU ORA

```

| | | | |
|------------|----------|-----------|--------------------------------------|
| 290 : C800 | | SEKUNDE = | ZEHNTTEL +1 |
| 300 : C800 | | MINUTE = | SEKUNDE +1 |
| 310 : C800 | | STUNDE = | MINUTE +1 |
| 320 : C800 | | = | STUNDE +3 ;50/60 HZ |
| 330 : C800 | | TRIGGER = | TRIGGER +1 ; ID0/EBRESZTO BEALLITASA |
| 340 : C800 | | *= | \$C800 |
| 350 : C800 | AD 0E DC | LDA | TRIGGER |
| 360 : C803 | 09 80 | ORA | #%10000000 ;50-HZ-ES MOD |
| 370 : C805 | 8D 0E DC | STA | TRIGGER |
| 380 : C808 | AD 0F DC | LDA | SET |
| 390 : C80B | 29 7F | AND | #%1111111 ;AZ ID0 BEALLITASA |
| 400 : C80D | 8D 0F DC | STA | SET |
| 410 : C810 | 20 79 00 | JSR | CHRGOT ;VAN-E TOVABBI KOD |
| 420 : C813 | F0 65 | BEQ | CHGIER ;AZ ORA KIKAPCSOLASA |
| 430 : C815 | 20 FD AE | JSR | CHKCOM |
| 440 : C818 | 20 9E AD | JSR | FRMEVL ;EGY FUZER BETOLTESE |
| 450 : C81B | 20 A3 B6 | JSR | FRESTR ;PARAMETER |
| 460 : C81E | C9 06 | CMP | #6 ;6 KARAKTER |
| 470 : C820 | D0 6B | BNE | ILL ;'ILLEGAL QUANTITY' |
| 480 : C822 | A0 00 | LDY | #0 |
| 490 : C824 | B1 22 | LDA | (ADR),Y |
| 500 : C826 | 38 | SEC | |
| 510 : C827 | E9 30 | SBC | #"0" ;ATVALTAS HEX-RA |
| 520 : C829 | C9 03 | CMP | #3 |
| 530 : C82B | B0 60 | BCS | ILL |
| 540 : C82D | 0A | ASL | |
| 550 : C82E | 0A | ASL | |
| 560 : C82F | 0A | ASL | |
| 570 : C830 | 0A | ASL | |
| 580 : C831 | 85 FB | STA | TEMP |
| 590 : C833 | C8 | INY | |
| 600 : C834 | B1 22 | LDA | (ADR),Y |
| 610 : C836 | 38 | SEC | |
| 620 : C837 | E9 30 | SBC | #"0" |
| 630 : C839 | C9 0A | CMP | #10 |
| 640 : C83B | B0 50 | BCS | ILL |
| 650 : C83D | 05 FB | ORA | TEMP |
| 660 : C83F | D0 04 | BNE | NOTNULL |
| 670 : C841 | A9 92 | LDA | #%10000000 + #12 ; 12 |
| 680 : C843 | D0 0F | BNE | SETSTD |
| 690 : C845 | C9 24 | CMP | #\$24 |
| 700 : C847 | B0 44 | BCS | ILL |
| 710 : C849 | C9 13 | CMP | #\$13 |
| 720 : C84B | 90 07 | BCC | SETSTD |
| 730 : C84D | 38 | SEC | |
| 740 : C84E | F8 | SED | |
| 750 : C84F | E9 12 | SBC | #\$12 |
| 760 : C851 | D8 | CLD | |
| 770 : C852 | 09 80 | ORA | #%10000000 ;PM BEALLITASA |
| 780 : C854 | 8D 0B DC | STA | STUNDE |
| 790 : C857 | 20 FD CB | JSR | GETS9 ;A PERCEK BEOLVABASA |
| 800 : C85A | 8D 0A DC | STA | MINUTE |
| 810 : C85D | 20 FD CB | JSR | GETS9 |
| 820 : C860 | 8D 09 DC | STA | SEKUNDE |
| 830 : C863 | A9 00 | LDA | #0 |
| 840 : C865 | 8D 08 DC | STA | ZEHNTTEL ;AZ ORA INDITASA |
| 850 : C868 | 20 79 00 | JSR | CHRGOT |
| 860 : C86B | F0 0D | BEQ | CHGIRQ |
| 870 : C86D | 20 FD AE | JSR | CHKCOM |
| 880 : C870 | 20 9E B7 | JSR | BETBYT ;SZIN |
| 890 : C873 | E0 10 | CPX | #16 |
| 900 : C875 | B0 16 | BCS | ILL |
| 910 : C877 | 8E A7 02 | STX | COLOR ;A SZINKOD TAROLASA |
| 920 : C87A | 78 | SEI | ;IRQ-VEKTOR FELCSERELESE |
| 930 : C87B | AD 14 03 | LDA | IRQ |
| 940 : C87E | 49 A1 | EOR | #< IRQVEC ^ TIMIRQ |
| 950 : C880 | 8D 14 03 | STA | IRQ |

| | | | | | | |
|--------|------|----------|-------------------|-----|--------------------|--------------------------|
| 960 : | C883 | AD 15 03 | | LDA | IRQ + 1 | |
| 970 : | C886 | 49 22 | | EOR | #> IRQVEC ^ TIMIRQ | |
| 980 : | C888 | 8D 15 03 | | STA | IRQ + 1 | |
| 990 : | C88B | 58 | | CLI | | |
| 1000 : | C88C | 60 | | RTS | | |
| 1010 : | C88D | 4C 48 B2 | ILL | JMP | ILLQUAN | |
| 1020 : | | | | | | |
| 1030 : | C890 | A5 FB | | LDA | PNT | |
| 1040 : | C892 | 48 | | PHA | | |
| 1050 : | C893 | A5 FC | | LDA | PNT+1 | ;A MUTATO TARDLASA |
| 1060 : | C894 | 48 | | PHA | | |
| 1070 : | C895 | AD 88 02 | | LDA | VIDEO | ;A VIDEORAM (FELSO BYTE) |
| 1080 : | C899 | 85 FC | | STA | PNT+1 | |
| 1090 : | C89B | A9 00 | | LDA | #0 | |
| 1100 : | C89D | 85 FB | | STA | PNT | ;MUTATO A VIDEORAM-RA |
| 1110 : | C89F | A0 1E | | LDY | #30 | ;30.OSZLOP |
| 1120 : | C8A1 | AD 08 DC | | LDA | STUNDE | |
| 1130 : | C8A4 | C9 12 | | CMF | #\$12 | |
| 1140 : | C8A6 | F0 11 | | BEQ | NULLUHR | |
| 1150 : | C8A8 | C9 80 | | CMF | ##80 | |
| 1160 : | C8AA | 90 0F | | BCC | STDOUT | ;AM |
| 1170 : | C8AC | 29 7F | | AND | ##1111111 | |
| 1180 : | C8AE | C9 12 | | CMF | #\$12 | |
| 1190 : | C8B0 | F0 09 | | BEQ | STDOUT | |
| 1200 : | C8B2 | F8 | | SED | | |
| 1210 : | C8B3 | 18 | | CLC | | |
| 1220 : | C8B4 | 69 12 | | ADC | #\$12 | |
| 1230 : | C8B6 | D8 | | CLD | | |
| 1240 : | C8B7 | D0 02 | | BNE | STDOUT | |
| 1250 : | C8B9 | A9 00 | | LDA | #0 | |
| 1260 : | C8BB | 20 D8 C8 | NULLUHR STDOUT | JSR | PRINT | ;AZ ORAK KIJELZESE |
| 1270 : | C8BE | AD 0A DC | | LDA | MINUTE | |
| 1280 : | C8C1 | 20 D8 C8 | | JSR | PRINT | ;A PERCEK KIJELZESE |
| 1290 : | C8C4 | AD 09 DC | | LDA | SEKUNDE | |
| 1300 : | C8C7 | 20 D8 C8 | | JSR | PRINT | ;A MASODPERCEK KIJELZESE |
| 1310 : | C8CA | AD 08 DC | | LDA | ZEHNTEL | |
| 1320 : | C8CD | 09 30 | | ORA | #"0" | |
| 1330 : | C8CF | 20 F3 C8 | | JSR | PRINT1 | ;A TIZEDEK KIJELZESE |
| 1340 : | C8D2 | 68 | | PLA | | |
| 1350 : | C8D3 | 85 FC | | STA | PNT+1 | |
| 1360 : | C8D5 | 68 | | PLA | | ;A MUTATO VISSZATOLTESE |
| 1370 : | C8D6 | 85 FB | | STA | PNT | |
| 1380 : | C8D8 | 4C 31 EA | | JMP | IRQVEC | ;AZ IRQ MEGVALTOZTATASA |
| 1390 : | C8DB | 48 | | PHA | | ;KIIRAS |
| 1400 : | C8DC | 29 F0 | PRINT | AND | ##11110000 | |
| 1410 : | C8DE | 4A | | LSR | | |
| 1420 : | C8DF | 4A | | LSR | | |
| 1430 : | C8E0 | 4A | | LSR | | |
| 1440 : | C8E1 | 4A | | LSR | | |
| 1450 : | C8E2 | 18 | | CLC | | |
| 1460 : | C8E3 | 69 30 | | ADC | #"0" | |
| 1470 : | C8E5 | 20 F3 C8 | | JSR | PRINT1 | |
| 1480 : | C8E8 | 68 | | PLA | | |
| 1490 : | C8E9 | 29 0F | | AND | ##1111 | |
| 1500 : | C8EB | 18 | | CLC | | |
| 1510 : | C8EC | 69 30 | | ADC | #"0" | |
| 1520 : | C8EE | 20 F3 C8 | | JSR | PRINT1 | |
| 1530 : | C8F1 | A9 3A | | LDA | #" : " | |
| 1540 : | C8F3 | 91 F8 | PRINT1 | STA | (PNT),Y | ;KARAKTER |
| 1550 : | C8F5 | AD A7 02 | | LDA | COLOR | |
| 1560 : | C8F8 | 99 00 D8 | | STA | FARBE,Y | ;ES A SZIN |
| 1570 : | C8FB | C8 | | INY | | |
| 1580 : | C8FC | 60 | | RTS | | |
| 1590 : | C8FD | C8 | GETS9 | INY | | |
| 1600 : | C8FE | B1 22 | | LDA | (ADR),Y | ;TIZEDES |
| 1610 : | C900 | 38 | | SEC | | |
| 1620 : | C901 | E9 30 | | SBC | #"0" | |

| | | | | | |
|--------|------|-------|------|-----|-------------------|
| 1630 : | C903 | C9 06 | | CMP | #6 |
| 1640 : | C905 | B0 B6 | ILL1 | BCS | ILL |
| 1650 : | C907 | 0A | | ASL | |
| 1660 : | C908 | 0A | | ASL | |
| 1670 : | C909 | 0A | | ASL | |
| 1680 : | C90A | 0A | | ASL | |
| 1690 : | C90B | B5 FB | | STA | TEMP |
| 1700 : | C90D | C8 | | JNY | |
| 1710 : | C90E | B1 22 | | LDA | (ADR),Y ;EGYSEGEK |
| 1720 : | C910 | 38 | | SEC | |
| 1730 : | C911 | E9 30 | | SBC | "0" |
| 1740 : | C913 | C9 0A | | CMP | #10 |
| 1750 : | C915 | B0 EE | | BCS | ILL1 |
| 1760 : | C917 | 05 FB | | DRA | TEMP |
| 1770 : | C919 | 60 | | RTS | |

A BASIC betöltőprogram:

```

0 REM **** P89. ****
1 :
2 :
100 FOR I=51200 TO 51481
110 READ X:POKE I,X:S=B+X:NEXT
120 DATA 173, 14,220, 9,128,141, 14,220,173, 15,220, 41
130 DATA 127,141, 15,220, 32,121, 0,240,101, 32,253,174
140 DATA 32,158,173, 32,163,182,201, 6,208,107,160, 0
150 DATA 177, 34, 56,233, 48,201, 3, 176, 96, 10, 10, 10
160 DATA 10,133,251,200,177, 34, 56,233, 48,201, 10,176
170 DATA 80, 5,251,208, 4,169,146,208, 15,201, 36,176
180 DATA 68,201, 19,144, 7, 56,248,233, 18,216, 9,128
190 DATA 141, 11,220, 32,253,200,141, 10,220, 32,253,200
200 DATA 141, 9,220,169, 0,141, 8,220, 32,121, 0,240
210 DATA 13, 32,253,174, 32,158,183,224, 16,176, 22,142
220 DATA 167, 2,120,173, 20, 3, 73,161,141, 20, 3,173
230 DATA 21, 3, 73, 34,141, 21, 3, 88, 96, 76, 72,178
240 DATA 165,251, 72,165,252, 72,173,136, 2,133,252,169
250 DATA 0,133,251,160, 30,173, 11,220,201, 18,240, 17
260 DATA 201,128,144, 15, 41,127,201, 18,240, 9,248, 24
270 DATA 105, 18,216,208, 2,169, 0, 32,219,200,173, 10
280 DATA 220, 32,219,200,173, 9,220, 32,219,200,173, 8
290 DATA 220, 9, 48, 32,243,200,104,133, 52,104,133,251
300 DATA 76, 49,234, 72, 41,240, 74, 74, 74, 74, 24,105
310 DATA 48, 32,243,200,104, 41, 15, 24, 05, 48, 32,243
320 DATA 200,169, 58,145,251,173,167, 2,153, 0,216,200
330 DATA 96,200,177, 34, 56,233, 48,201, 6,176,134, 10
340 DATA 10, 10, 10,133,251,200,177, 34, 56,233, 48,201
350 DATA 10,176,238, 5,251, 96
360 IF S<32970 THEN PRINT "HIBA A DATA SORBAN..!":END
370 PRINT "OK..!!"

```

A saját BASIC program elindítása előtt az órát a következő utasítással állíthatjuk be:

SYS 51200 , "HHMMSS" , SZIN

ahol a HHMMSS az óra pillanatnyi állását (óra, perc, másodperc), a SZIN pedig a kijelzés színének kódját (0-tól 15-ig) jelenti. Ha például 14 óra 30 perc 15 másodperc az óra pillanatnyi állása, és sárga színű kijelzést szeretnénk, a szükséges utasítás:

SYS 51200, "143015",7

Az óra kikapcsolása: SYS 51200. Az utasítás hatására ugyan a kijelzés szünetel, de az óra számlálása tovább folyik. A következő SYS 51200 utasítás hatására az aktuális idő ismét megjelenik a képernyőn.

Most néhány tanács a „multiprogramozáshoz”.

A második job (munka) elsővel párhuzamos végrehajtására két lehetőség van.

Az első lehetőség a megszakítóprogram felhasználása, amelyet a rendszer minden hatvanad másodpercben meghív. Ezt a megoldást használtuk a bemutatott órakijelző programban.

A megszakítási vektor címét megváltoztattuk úgy, hogy az a saját rutinunkra mutasson, amelyet viszont egy, az eredeti megszakító rutinra ugrással zártunk azért, hogy közben a szokásos és egyben szükséges billentyű-lekérdezés se maradjon el.

A második megoldásban a saját rutint úgy írjuk meg, hogy az maga idézze elő a megszakítást.

Ha például egy szöveget akarunk kinyomtatni programfutás közben, a nyomtató BUSY vezetéket használhatjuk a megszakítás vezérlésére.

Valahányszor a nyomtató készen áll egy karakter kiírására, ez megszakítást vált ki. A megszakító rutin elküldi a karaktert a nyomtatóhoz, végrehajtja a saját feladatát, majd visszatér a főprogramra. Amint a nyomtató készen van az elküldött karakter feldolgozásával, ismét megszakítás következik stb. A felhasználó az időosztást nem fogja érzékelni. Ahhoz, hogy valaki egy ilyen gépi kódú rutint el tudjon készíteni, nagyon alaposan ismernie kell a Commodore 64-es operációs rendszerét, amelyet „A C 64 belső felépítése” című könyv teljes mélységében bemutat Olvasóinak.

9.7 A POKE utasítások és a nulláslap

A programozás során az Olvasó biztosan tapasztalta, hogy mind a BASIC, mind pedig a gépi nyelven írt programokban gyakran szükség van bizonyos tárcímek tartalmára, vagy annak módosítására. Ezért ismertetjük a legfontosabb tárcímek rendeltetését:

| <i>Cím</i> | <i>Feladat</i> |
|------------|---|
| 0000–0001 | Ha valamelyik vagy mindkét cím tartalmát módosítjuk (POKE utasítással), egy meghatározott tárterületet ki/bekapcsolhatunk. |
| 0043–0044 | A felhasználói tár, azaz a BASIC terület kezdőcímét tartalmazó tárcímek. A BASIC kezdet leolvasása: $PEEK(43) + 256 * PEEK(44)$. POKE utasítással tetszőlegesen módosíthatjuk a BASIC kezdetet. |
| 0045–0046 | A numerikus változókat tartalmazó tárterület kezdete. Alapállapotban ez a tárterület a BASIC tár alatt helyezkedik el. |
| 0047–0048 | Az indexes változókat tartalmazó tárterület kezdete |
| 0049–0050 | Az indexes változókat tartalmazó tárterület vége |
| 0051–0052 | A BASIC program füzérváltozóinak kezdete |
| 0055–0056 | A BASIC RAM végcíme. Megváltoztatva tartalmukat, „megvédhetünk” bizonyos területeket a BASIC programtól (pl. gépi kódú rutinokat) és emellett szabadon rendelkezhetünk a \$C000-tól \$CFFF-ig terjedő tárterülettel. Pl. a POKE 55,0 : POKE 56,64 utasításokkal a BASIC RAM végét a \$4000 tárcímre helyezhetjük. |
| 0115–0138 | Ezek a tárcímeken található a CHRGET rutin, amely a BASIC területről betölt egy karaktert. Ha BASIC bővítéssel akarunk dolgozni, ezt a rutint kell megváltoztatnunk. |
| 0203 | Az éppen lenyomott billentyű kódját tartalmazó tárcím. Ha ezen a címen 64 áll, egyetlen billentyű sincs lenyomva. |

KIEGÉSZÍTÉS

Az alábbi két sor hiányzik a P. 71-es programból:

3003 DV = 1

3004 GOSUB 8000:GOSUB 8500

JEGYZET

TOVÁBBI DATA BECKER KÖNYVEK A NOVOTRADE RT. KIADÁSÁBAN

A Commodore 64 belső felépítése

A könyv belülről világítja meg az Ön kedvenc számítógépét. Nélkülözhetetlen mindazok számára, akik a mikrogép rejtelmeinek mélyére akarnak ásni. A kötet a C 64-es minden szempontból alapos leírását adja. Teljes ROM listát tartalmaz részletes magyarázatokkal felszerelve, így Ön kedve szerint tanulmányozhatja a BASIC interpreter, a kernal vagy az operációs rendszer működését. Számos példát talál majd a gépi nyelvű programozásra és továbbiakat, melyek révén az Ön programozói munkája kellemesebb lesz.

Gépi kódú programozás a Commodore 64-esen

Azok számára íródott, akik már nem a BASIC nyelven, hanem a gyorsabb, a memóriát hatékonyabban kihasználó gépi kódban kívánnak programozni. A könyv kifejezetten a C 64-eshez íródott. Tanulja meg, hogyan működik a 6510-es mikroprocesszor a C 64-esen. A ROM rutinok hozzáférhetősége, I/O, BASIC bővítés stb. Tartalmazza 3 teljes program listáját: egy ASSEMBLER-ét, egy DISASSEMBLER-ét és egy nem mindennapi 6510 SIMULATOR-ét, melynek segítségével Ön valóban „látni fogja” működés közben a C 64-est.

A VC 1541-es lemezegység programozása

A könyv a 1541-es lemezegység minden titkára fényt derít. Az első fejezetekben a program-, a soros és a relatív file-okat, ill. az ezekkel történő adattárolás módjait tárgyalja. Ezután bemutatja a közvetlen elérésű utasításokat, a lemez felépítését, a DOS működési elvét. A részletesen kommentált teljes DOS listát az „inyencek” figyelmébe ajánljuk. Szerepel a könyvben néhány hasznos segédprogram (a BACKUP, a COPY, a DIRECTORY stb.). Mindent egybevetve, a kötet a 1541-es lemezegység alapvető kézikönyve.

Ára: 302,— Ft

Értesítjük vevőinket, üzleti partnereinket, hogy

2c

SZÁMÍTÁSTECHNIKAI ÁRUHÁZ

néven megnyitottuk üzletünket.

AZ ÁRUHÁZ SZOLGÁLTATÁSAI:

- ◆ számítástechnikai rendszerek komplex leasingje
- ◆ komplett rendszerek kulcsrakész átadása, betanítása
 - ◆ meglévő szoftverek átírása
- ◆ oktatás ◆ olvasószolgálat ◆ szaktanácsadás
- ◆ alkalmazói szoftverek igény szerinti kialakítása
- ◆ felhasználóknak, user kluboknak helyszíni gépidő biztosítása

AZ ÜZLET NYITVATARTÁSI IDEJE:

Hétfőtől — Péntekig 9—18 óráig

AZ ÜZLET CÍME:

1136 Budapest, Balzac u. 35. Tel.: 402-954

NOVOTRADE